UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO DE FIN DE GRADO

Grado en Ingeniería Biomédica

REDES GENERATIVAS ANTAGÓNICAS PARA LA ESTANDARIZACIÓN DE IMÁGENES DE CÉLULAS DE SANGRE PERIFÉRICA

Memoria y Anexos

Autor: Alejo Pérez Gómez
Director: Rodellar Benedé, José Julián
Co-Director: Alférez Baquero, Edwin Santiago

Convocatoria: 2019 - 2020

Resum

L'anàlisi de sang perifèrica mitjançant l'anàlisi de frotis és una eina útil per a la detecció de malalties i anomalies. Ja sigui a través de microscòpia o gràcies a l'ús de maquinària automàtica especialitzada, es poden dur a terme molts exàmens de fluids o segregacions corporals de gran utilitat diagnòstica. Com és en el cas d'aquest treball, l'estudi o anàlisi de cèl·lules hematològiques que circulen en sang perifèrica requereix d'una tinció específica a l'hora de millorar el contrast i la visibilitat en el microscopi manual o motoritzat com el Cellavision DM96.

L'equip de CellsiLAB, grup d'investigació resultant de la col·laboració de l'Hospital Clínic i la Universitat Politècnica de Catalunya, ha desenvolupat models de xarxes neuronals per dur a terme classificacions automàtiques de les cèl·lules sanguínies anòmales segons el seu tipus (limfòcits anormals en diferents tipus de limfoma, blasts en leucèmies agudes, limfòcits reactius en infeccions, entre d'altres casos). Per altra banda, ocorre que les mostres obtingudes en centres hospitalaris aliens al Hospital Cínic són més susceptibles de ser classificades erròniament per aquests models al no compartir característiques de color, tinció i textura.

És aquí on entren les GAN, traduïdes al català com a Xarxes Generatives Antagòniques, uns tipus de xarxes neuronals artificials que han experimentat una gran evolució en els darrers anys. La particularitat d'aquestes arquitectures és que es generen imatges a partir de l'aprenentatge d'un conjunt de dades.

Per això, s'ha aprofitat aquesta qualitat per transformar les imatges del domini X (Hospital Germans Trias i Pujol) en el domini Y (Hospital Clínic) i així emular les característiques de color, tinció i textura d'aquesta última. Gràcies a aquesta transformació s'ha aconseguit millorar l'efectivitat en la classificació de les imatges fetes a l'Hospital Germans Trias i Pujol per part dels models de xarxes neuronals convolucionals pre-entrenades amb conjunts de dades de l'Hospital Clínic.

Per a les transformacions s'han utilitzat les arquitectures *CycleGAN* i *ColorizationGAN*, mentre que per a les classificacions s'han usat: ResNet 18, ResNet 34, ResNet 18 amb Focal Loss i ResNet 34 amb Focal Loss. S'ha aconseguit una millora de la classificació del 34% al 81%.

Paraules clau: Blasts, GANs, CNN, leucocitosi, Limfòcits, CycleGAN, ColorizationGAN

Resumen

El análisis de sangre periférica mediante análisis del frotis es una útil herramienta para la detección de enfermedades y anomalías. Ya sea mediante microscopía o gracias al uso de maquinaria automática especializada, se pueden llevar a cabo numerosos exámenes de fluidos o segregaciones corporales de gran utilidad diagnóstica. Como en el caso de este trabajo, el estudio o análisis de células hematológicas que circulan en sangre periférica requiere de una tinción específica a la hora de mejorar el contraste y visibilidad en el microscopio manual o motorizado como el Cellavision DM96.

El equipo de CellsiLAB, grupo de investigación fruto de la colaboración del Hospital Clínic y la Universidad Politécnica de Cataluña, ha desarrollado modelos de redes neuronales para llevar a cabo clasificaciones automáticas de las células sanguíneas anómalas según su índole (linfocitos anormales en diferentes tipos de linfoma, blastos en leucemias agudas, linfocitos reactivos en infecciones, entre otros casos). Asimismo, ocurre que las muestras obtenidas en hospitales ajenos son más susceptibles a ser clasificadas erróneamente por dichos modelos al no compartir características de color, tinción y textura.

Es aquí donde entran las GANs, traducido al español RGAs, Redes Generativas Antagónicas, un tipo de redes neuronales artificiales que han experimentado una enorme evolución en los últimos años. La particularidad de estas arquitecturas es que generan imágenes a partir del aprendizaje de un conjunto de datos.

Por ende, se ha aprovechado esta cualidad para transformar las imágenes del dominio X (Hospital Germans Trias i Pujol) en el dominio Y (Hospital Clínic) y así emular las características de color, tinción y textura de este último. Tras esta transformación se ha conseguido mejorar la efectividad en la clasificación de las imágenes tomadas en el Hospital Germans Trias i Pujol por parte de los modelos de redes neuronales convolucionales pre-entrenados con *datasets* del Hospital Clínic.

Para las transformaciones se han utilizado las arquitecturas *CycleGAN* y *ColorizationGAN,* mientras que para las clasificaciones se han utilizado: ResNet 18, ResNet 34, ResNet 18 con *focal loss* y ResNet 34 con *focal loss.* Se ha logrado mejorar la exactitud de la clasificación *del* 34% a al 81%.

Palabras Clave: Blastos, GANs, CNN, Leucocitos, Linfocitos, CycleGAN, ColorizationGAN

Abstract

Peripheral blood testing by smear analysis is a useful tool for the detection of diseases and abnormalities. Whether by microscopy or thanks to the use of specialized automatic machinery, numerous examinations of body fluids or secretions of great diagnostic value can be carried out. As in the case of this work, the study or analysis of haematological cells circulating in peripheral blood requires specific staining to improve contrast and visibility on the manual or motorised microscope such as the Cellavision DM96.

The CellsiLAB team, a research group that is the result of collaboration between Hospital Clínic and the Polytechnic University of Catalonia, has developed models of neuronal networks to carry out automatic classifications of abnormal blood cells according to their nature (abnormal lymphocytes in different types of lymphoma, blasts in acute leukaemias, reactive lymphocytes in infections, among other cases). Nevertheless, those samples obtained in other hospitals are more susceptible to be misclassified by such models because they do not share characteristics of color, staining and texture.

This is where GANs come in, a type of artificial neural networks that have undergone an enormous evolution in recent years. The particularity of these architectures is that they generate images by means of the machine learning of a set of data.

Therefore, this quality has been used to transform the images of the X domain (Hospital Germans Trias i Pujol) into the Y domain (Hospital Clínic) emulating characteristics of colour, staining and texture of the latter. After this transformation, the effectiveness of the classification of the images taken at Hospital Germans Trias i Pujol has been improved with the use convolutional neural network models pre-trained with Hospital Clínic datasets.

CycleGAN and ColorizationGAN architectures have been used for the transformations, while ResNet 18, ResNet 34, ResNet 18 with Focal Loss and ResNet 34 with Focal Loss have been used for the classifications. The classification accuracy has been improved from 34% to 81%.

Keywords: Blasts, GANs, CNN, Leukocytes, Lymphocytes, CycleGAN, ColorizationGAN

Agradecimientos

Después de 5 meses de intenso trabajo, me gustaría agradecer a las personas del grupo CellsiLAB por toda la ayuda y guía recibidas a lo largo de este viaje. Gracias a mi director José Rodellar por haberme brindado esta oportunidad. Gracias a la Dra. Anna Merino por su gran ayuda y darme acceso a las imágenes de los hospitales (material indispensable para este TFE), al igual que a Andrea Acevedo, Laura Boldú y Cristian Morales.

Por último, quisiera agradecer a mi codirector Santiago Alférez por su incansable asesoramiento y ayuda.

Glosario

ML: Machine Learning – Aprendizaje Automático

Deep Learning: Aprendizaje Profundo

GAN: Generative Adversarial Network – Redes Generativas Antagónicas

LR: Learning Rate – Ratio de Aprendizaje, Parámetro que marca la velocidad a la que aprende la red neuronal

CNN o ConvNet: Convolutional Neural Network - Red Neuronal Convolucional, donde el operador matemático básico es la convolución

ReLU: Rectified Linear Unit – Unidad Lineal Rectificada, función que convierte los negativos de una distribución de datos a 0.

White Blood Cells (WBC) – Glóbulos Blancos (GB) – Leucocitos

Red Blood Cells (RBC) – Glóbulos Rojos (GR)– Eritrocitos – Hematíes

Sangre Periférica – SP

Dataset – Conjunto de Datos, ya sea para el entrenamiento, validación o test de una RN

IA – Inteligencia Artificial

Fine tuning – Ajuste fino de la red mediante entrenamiento aprovechando el conocimiento que adquirió en un pasado el modelo de red neuronal

Dataroot – Directorio de localización de los datos para procesar

Índice

1. Prefacio

1.1. Origen del trabajo

Este trabajo nace de la necesidad de compatibilizar imágenes citológicas de diferentes centros sanitarios con la base de datos de imágenes del Hospital Clínic, concretamente de la serie leucocitaria o serie blanca. Esto es debido a que la precisión de las redes neuronales clasificadoras del Hospital Clínic disminuye al clasificar imágenes de otros hospitales. Esto se debe a que los modelos clasificadores del Clínic se han entrenado con un *dataset* de características de color, tinción y textura diferentes a los *datasets* de otros hospitales.

Es necesario, por lo tanto, crear un algoritmo de pre-procesamiento para tratar de emular las características de tinción de las imágenes tomadas en el Hospital Clínic por parte de imágenes ajenas a éste. Así se pretende asegurar la convergencia en el entrenamiento de los modelos y una mayor precisión en cuanto a la producción de la respuesta correcta (mejor clasificación de imágenes de las células).

El proyecto tiene fines similares a los de la tesis de máster *"Cell Image Transformation Using Deep Learning"* [1], presentada en la Universidad de Lund en colaboración con la compañía Cellavision. En dicha tesis se utilizan *GANs* para la estandarización de imágenes de células sanguíneas de la serie blanca para emular las características de captación de imagen de un modelo de analizador Cellavision en concreto.

1.2. Motivación

La motivación de este trabajo se remonta al interés que creció en mi después de realizar diferentes cursos de *data science* aplicada mediante Python, que se vio incrementado tras cursar la asignatura de "Aprendizaje Bioestadístico". Dicha asignatura, impartida por mis directores de TFE (José Rodellar y Santiago Alférez) me mostró el enorme potencial del *machine learning* a nivel de la visión por computador, además de sus otras vertientes, hecho que me llevó a encauzar mi carrera por este camino y evolucionar en esta materia.

1.3. Requerimientos previos

Para realizar este proyecto, es necesario tener una experiencia notable en programación orientada a objetos y *data science* con el lenguaje de programación *Python*. Además, se requiere haber realizado el curso de *Fastai*: *Deep Learning for Coders* (Parte 1) y parte del de *Deep Learning from the Foundations* (Parte 2).

También ha sido oportuno el aprendizaje de la librería *Pytorch*, orientada al cálculo tensorial, además de otras librerías como *Scikit-learn* (*data science* y estadística), *Os* y *Regural Expressions* (para recorrer directorios y manejo de archivos por código), *Pandas* (manejo de estructuras de datos tabulares o *dataframes*), *Matplotlib* (realización de gráficos), *Numpy* (cálculo numérico), etc.

En cuanto al *set-up* técnico a nivel *hardware* se ha contado con:

- Equipos: *iMac* de finales del 2013 / *Sony Vaio*
- Sistemas operativos: *MacOS Catalina 64 bits/ Windows 7 Home Premium* 64 bits
- Procesadores: *2,7 GHz Intel Core i5* de 4 núcleos / *Intel(R) Core (TM)2 Duo CPU @2GHz*
- Memoria *RAM*: *8GB / 4 GB*
- Tarjetas gráficas: *Intel Iris Pro 1536 MB / Intel® 815 EM Integrated Graphic*
- Estación de trabajo: *DeepBox* con *GPU Titan XP (12 GB)*

En cuanto al *set-up* técnico a nivel *software* se ha contado con:

- *Python 3.6.8*
- *Conda 4.5.12*
- *Pytorch 1.0.0*
- *Fastai 1.0.54*
- *Pandas 0.24.1*
- *Matplotlib 3.0.2*
- *Shutil 1.7.0*
- *Os*
- *Numpy 1.15.4*
- *Sklearn 0.21.2*
- *Regex 2018.1.10*

Para otros aspectos técnicos, ha sido necesario la instalación y aprendizaje de los siguientes *softwares open source*:

- Aprender a utilizar la consola de *MAC* (basada en *Unix*)
- Conexión a *Deepbox* mediante *SSH (*gracias al *software PUTTY)* y *VPN* (gracias al *software F5 VPN*)
- *Window Manager Screen GNU* (permite realizar procesos como entrenamientos de red neuronal con desconexión de la VPN), etc.
- *Github* (servicio de repositorio software online donde se ha almacenado código y se han controlado las versiones de las *Notebooks*)

1.4. Planificación del trabajo

Para planificar este trabajo se ha utilizado el servicio web ***Trello*** (un tablero *Kanban* online) con el que se han podido organizar las tareas. A continuación, se procede a describir brevemente cada proceso (fase) que figura en el diagrama Gantt de este apartado (Figura 1.1).

1. La primera fase del trabajo ha constado de la realización de los dos cursos de *Fastai* y de la lectura de 12 *artículos* relativos a las *Generative Adversarial Networks.*
2. En la segunda fase se han investigado las tecnologías o arquitecturas con las que poder realizar el cometido de este trabajo (diferentes tipos de GAN, librerías de programación, etc.). Al final se ha optado por las arquitecturas *CycleGAN* y la *ColorizationGAN.*
3. La tercera fase corresponde a un estudio realizado sobre las metodologías de *machine learning* a emplear (utilización de redes neuronales convolucionales, funciones de pérdida, *Backpropagation*, métricas, entrenamiento de CNNs, etc.).
4. Similar a la tercera, la cuarta fase ha consistido en estudiar las bases biológicas y citológicas que sustentan el TFE (sistema inmunitario, sistema linfático, componentes celulares de distinto tipo, anomalías de leucocitos, etc.).
5. En la quinta fase se ha aprendido a utilizar la consola *Unix* para trabajar con *MAC* y la consola *CMD* para trabajar con el PC que utiliza *Windows*. Esto ha sido necesario para establecer la conexión con la *deepbox*, el movimiento por los directorios de la estación de trabajo, ejecución de los scripts que accionan el entrenamiento y *test* de *GANs*, etc.
6. En la sexta fase se han realizado los primeros ensayos de clasificación con CNNs en el conjunto del Hospital Clínic, realizando el *fine tuning* de las redes clasificadoras con este *dataset.*
7. En la séptima fase se ha llevado a cabo la conversión del *dataset* del Hospital Germans Trias i Pujol a los estándares del Hospital Clínic por medio de las GANs escogidas. Este proceso se incluye dentro del Bloque 1 de la metodología.

8. En la octava fase se han realizado las clasificaciones por CNN de las variantes del conjunto del Hospital Germans Trias i Pujol (tanto las generadas como la variante original). Estos experimentos corresponden al Bloque 2 de la metodología.
9. Correspondiendo con el Bloque 3 de metodología de este TFE, en la novena fase se han ejecutado las transformaciones entre clases de linfocito del *dataset* del Hospital Clínic.
10. En la décima fase se han agrupado, tabulado, analizado y discutido los resultados obtenidos en los tres bloques de metodología. Estos resultados corresponden a métricas de desempeño de los modelos clasificadores y a consideraciones cualitativas con respecto a las imágenes generadas por GANs.
11. La undécima fase no se ha realizado al acabar todas las anteriores, sino que ha sido un proceso continuo a lo largo de todo el período de realización de este proyecto. Este representa el redactado de la memoria y se ha llevado a cabo conforme se realizaban las fases anteriores.

Duración	Nombre de la tarea	Inicio	Finalizar
12d	1 - Cursos de Fastai	21/08/19	05/09/19
8d	2 - Lectura de papers sobres GANs	28/08/19	06/09/19
10d	3 - Investigación de tecnologías de ML	05/09/19	18/09/19
8d	4 - Investigación sobre bases citológicas	16/09/19	25/09/19
6d	5 - Aprendizaje consola UNIX (MAC)	11/09/19	18/09/19
12d	6 - Pruebas clasificación (Fine Tune Clínic)	26/09/19	11/10/19
18d	7 - Transformación GANs conjunto Can Ruti (1r Bloque)	08/10/19	31/10/19
16d	8 - Clasificación conjuntos Can Ruti (2o Bloque)	01/11/19	22/11/19
12d	9 - Transformaciones clases del Clínic (3er bloque)	25/11/19	10/12/19
13d	10 - Recogida de resultados	11/12/19	27/12/19
64d	11 - Redactado de la Memoria	07/10/19	02/01/20

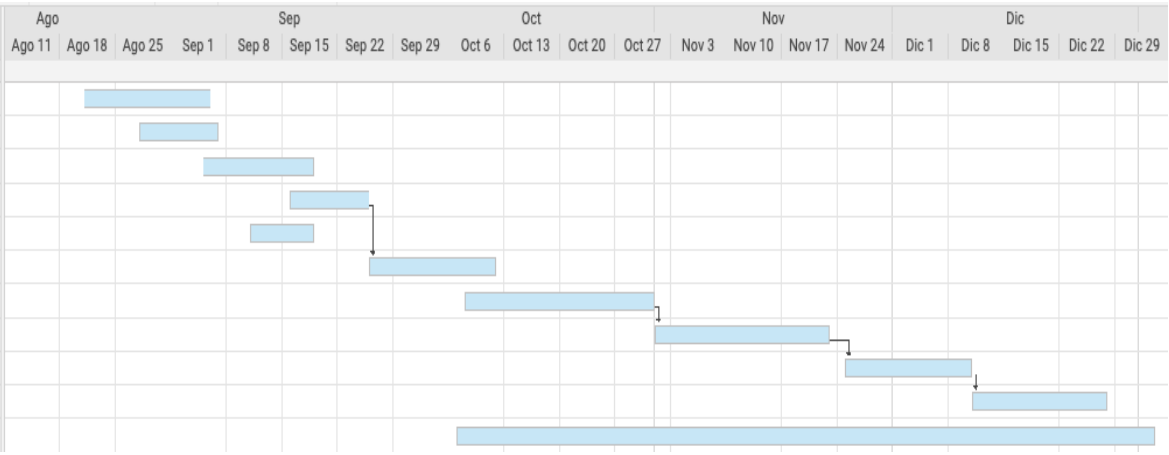


Figura 1.1. Diagrama de Gantt de la planeación del TFE. Fuente: Elaboración propia.

2. Introducción

El *Machine Learning* y más concretamente el *Deep Learning*, son campos de estudio que avanzan desenfrenadamente estos últimos años. Esto es gracias a que muchos descubrimientos y mejoras en estadística, ciencias de la computación u otros sectores repercuten en el progreso del aprendizaje automático. Estas mejoras en el aprendizaje ayudan a progresar en el resto de disciplinas científicas, donde se busca aplicar cada vez más el uso de herramientas de IA.

En el campo de la observación citológica, como es el de este TFE, ya son numerosos estudios los que han implementado algoritmos de *Machine Learning* para diferentes tareas, como por ejemplo la clasificación automática de células según su tipo, extracción de características, aprendizaje no supervisado y demás.

Este trabajo en concreto plantea la utilización de las RNs más novedosas hasta el momento (las GANs) para la estandarización de imágenes de células de sangre periférica previa a su posterior clasificación automática. Este paso de pre-procesado tiene como objetivo el aumento de efectividad de los modelos clasificadores de RNs utilizados por el grupo CellsiLAB cuando estos se enfrentan a imágenes tomadas en un hospital ajeno al Clínic.

En los Capítulos 3 y 4 se presentan las bases biológicas y de *Machine-Deep Learning* respectivamente para establecer un conocimiento de fondo y entender así la metodología del proyecto.

El proyecto constará de tres grandes bloques metodológicos, estrechamente relacionados con los objetivos. El primero consistirá en la utilización de GANs para la conversión de las imágenes de células de sangre periférica (SP) del Hospital Germans Trias i Pujol a imágenes que compartan características de tinción, color y textura con las de la base de datos del Hospital Clínic.

El segundo bloque tratará de clasificar automáticamente estas imágenes transformadas para comprobar la utilidad de los procesos del primer bloque, para ello se utilizarán RNs Convolucionales.

En el tercer bloque, al margen de los dos anteriores, se utilizarán las GANs para transformar células del Hospital Clínic de una clase a otra (por ejemplo: de linfocitos normales a blastos) por medio del aprendizaje automático y la inteligencia artificial.

Para ver los códigos, resultados de experimentos y *Jupyter Notebooks* de este proyecto se facilita el repositorio de *Github* donde se han almacenado. La URL es la siguiente:

`https://github.com/alejo-perez-upc-77/TFG/tree/master/Notebooks_Results_TFG` [2].

2.1. Objetivos del trabajo

1. **Bloque 1.** Utilizar Redes Neuronales Generativas Antagónicas para transformar imágenes de células de sangre periférica procedentes del *Hospital Germans Trias i Pujol* (referido de aquí en adelante como Hospital de *Can Ruti*) en imágenes similares a las obtenidas en el *Hospital Clínic i Provincial de Barcelona* (referido de aquí en adelante como *Hospital Clínic*).
   1.1. Crear un algoritmo de pre-procesado de imágenes citológicas de la serie blanca para su posterior clasificación automática mediante Redes Neuronales Convolucionales. Se diferenciarán los dominios *X* e *Y* como dos distribuciones de imágenes de células tomadas en Can Ruti y Clínic respectivamente, cada una con sus respectivos estándares de color, tinción y textura. El objetivo es entrenar dos arquitecturas diferentes (*CycleGAN* y *ColorizationGAN,* basada en la arquitectura *Pix2Pix*) para realizar la conversión de X a Y.
   1.2. Valorar cualitativamente los resultados de generación de imágenes (*fake)* comparándolas con las originales del Clínic. Comparar los conjuntos imágenes generadas (*fake)* por ambas arquitecturas entre ellos y con el conjunto Can Ruti original.

2. **Bloque 2.** Clasificar de forma automática las imágenes pertenecientes al Hospital Can Ruti (originales y generadas) utilizando redes CNN clasificadoras. Este proceso constará de tres partes:
   2.1. Realizar un *fine tuning* (ajuste fino de la red mediante entrenamiento) con un conjunto de imágenes del Clínic en CNNs pre-entrenadas con la base de datos pública *Imagenet* [3]. En algunos casos se modificará la función de pérdida de la red.
   2.2. Clasificar automáticamente, mediante las CNNs descritas en el punto 2.1, las imágenes del hospital Can Ruti (las generadas mediante GAN y las originales). Posteriormente, valorar los resultados y comparar los desempeños de las redes en los conjuntos de Can Ruti (originales y *fake*).
   2.3. Realizar otro *fine tuning* con estas CNNs utilizando un set de entrenamiento de los conjuntos de imágenes de Can Ruti (cada uno de los diferentes *fake* y el original) y repetir la clasificación automática en los respectivos *datasets.* Posteriormente, valorar los resultados y comparar los desempeños de las redes en todos los conjuntos de Can Ruti.

3. **Bloque 3.** Generar de forma artificial imágenes de células anormales de sangre periférica a partir de células sanguíneas linfoides normales del Hospital Clínic y viceversa mediante *CycleGAN*.
   3.1. Seleccionar el conjunto de imágenes de células linfoides normales y diferentes conjuntos de imágenes de células anormales de distinta índole para entrenar la *CycleGAN*.

3.2. Generar y valorar cualitativamente las imágenes obtenidas. Se harán experimentos para distintas patologías.

2.2. Alcance del trabajo

El alcance de este trabajo está constreñido por la densidad y cantidad de la base de datos de imágenes con la que se podrá trabajar, siendo las imágenes biológicas de pacientes de difícil acceso. Esto es debido a la ley de protección de datos, que intenta salvaguardar la identidad y privacidad de los usuarios del hospital. Dicho lo cual, se efectuarán los procesos para poder tratar las imágenes facilitadas por la Dra. Anna Merino del Hospital Clínic, garantizando la privacidad y anonimizando en todo momento los recursos del trabajo.

Por otra parte, la práctica con redes neuronales actualmente se encuentra en un estado de continua experimentación y según los estudios previos publicados y estudiados, las metodologías de trabajo tienden a ser bastante heurísticas para obtener los resultados deseados. Es por esto que se espera un comportamiento del sistema estocástico en vez de determinista, ya que no hay certidumbre alguna de cómo van a ser los resultados.

Dado que los tiempos de entrenamiento de las GANs son bastante elevados, pudiendo durar días, el alcance del trabajo se ajustará a un TFE, ya que las limitaciones de rendimiento de hardware se limitan a los de una sola GPU, mientras que los estudios del estado del arte cuentan con más de cuatro pudiendo llegar a superar las diez unidades en paralelo.

Con respecto a los modelos GAN utilizados, no se realizarán modificaciones estructurales a las arquitecturas ya que eso excedería el alcance de un TFE.

2.3. Estado del arte — Observación digital de células de sangre periférica y Redes Neuronales

Varios estudios evalúan el desempeño de la pre-clasificación automática por tipos de leucocitos realizada por equipos motorizados de microscopía de SP. También se evalúan distintas métricas como el recuento celular. Siempre enfatizando la necesidad de validación de médicos expertos con microscopía manual para revisar la clasificación de los equipos y concretar diagnósticos o resultados vinculantes a pacientes [4].

Un ejemplo de microscopio motorizado que realiza preclasificaciones de leucocitos es el Cellavision DM96, evaluado por Merino et al. (2011) [5]. Este obtuvo buenos valores de exactitud, generalmente para los leucocitos y en menor medida para linfocitos reactivos, eritroblastos y metamielocitos, peores aún para linfocitos anormales y blastos que circulan en la sangre en enfermedades hematológicas malignas [4].

Sjöstrand y Jönsson (2018) [1] argumentan que el modelo más comercializado de Cellavision es el modelo *Sysmex DI-60*, dirigido para laboratorios de alto rendimiento. Al no comercializarse más el modelo DM96, toman el relevo el DM9600 y DM1200, con iluminaciones diferentes por efecto del cambio de lámparas en su tecnología. Su trabajo trata de utilizar Deep Learning para conseguir una conversión mediante GANs desde imágenes de DM1200 a DM96, para así ahorrarse costosos pasos de pre-procesado y poder clasificarlas con las redes neuronales pre-entrenadas existentes.

Respecto a la clasificación linfocítica mediante RNs, *T. Pansombut et al. (2019)* [6], *J. Zhao et al. (2018)* [7], *B. Hu et al.* (2017) [8] y *L. Boldú et al. (2019)* [9] proponen clasificadores automáticos de diferentes tipos de glóbulos blancos mediante CNN (en combinación con *Decision Trees* y *Support Vector Machine* en el caso de *J. Zhao et al. (2018)*). *T. Pansombut et al. (2019)* utilizan CNNs para la clasificación de blastos linfoides de diferentes subtipos. *S. Alférez* (2015) [4] propone diversos algoritmos de inteligencia artificial y procesado de imagen para clasificación y segmentación de células linfoides normales y neoplásicas de distinta índole.

Relacionado con el uso de GANs en clasificación de leucocitos, *B. Hu et al. (2017)* proponen el uso de estas arquitecturas como complemento al aprendizaje no supervisado para la representación visual de histopatologías.

En lo que respecta a las GANs en general, se destaca la Style-GAN [10] como el actual estado del arte en generación de imagen con control del espacio latente produciendo imágenes con ultra resolución variable mediante la modificación de "estilo" (posturas e identidad aplicado a rostros humanos). En cuanto a generación de imagen per se, BigGAN [11] se postula como la arquitectura de referencia entrenada en la *database Imagenet*.

Es importante mencionar las arquitecturas *CycleGAN* [12] (utilizada en este proyecto) y *Pix2Pix* [13] (en la que se basa la *ColorizationGAN*, también utilizada en este TFE). En [12] se propone la transformación semi-supervisada (en términos de estilo) de imágenes de un conjunto *A* en un conjunto *B*. Por su parte, [13] utiliza las arquitecturas *U-Net* y *Resnet* como generadores sumado a la introducción de una función de coste por parches en su discriminador, el *PatchGAN.*

3. Bases Biológicas

3.1. La sangre

La sangre, al igual que todos los tejidos conectivos, está formada por componentes celulares y una matriz extracelular. Estos elementos celulares incluyen eritrocitos o glóbulos rojos (RBC ó GR), leucocitos o glóbulos blancos (WBC ó GB) y fragmentos de células llamados plaquetas. La matriz extracelular, llamada plasma, hace que la sangre sea única entre los tejidos conectivos porque es fluida. Este fluido, que está constituido principalmente por agua, suspende perpetuamente los elementos y les permite circular por todo el cuerpo dentro del sistema cardiovascular [cap. 18.1, 14].

3.2. Las células sanguíneas

3.2.1. Eritrocitos, hematíes o glóbulos rojos

La función principal de los GR es transportar oxígeno a las células del cuerpo y llevar dióxido de carbono a los pulmones. Su forma bicóncava ayuda a la capacidad de los glóbulos rojos de maniobrar a través de pequeños vasos sanguíneos para entregar oxígeno a los órganos y tejidos.

Aumentando así su relación superficie/volumen, permiten que el O_2 y el CO_2 se difundan más fácilmente a través de la membrana plasmática de los glóbulos rojos. Esta molécula porta consigo hierro que se une al oxígeno a medida que las moléculas de oxígeno entran en los vasos sanguíneos de los pulmones. Dicha molécula también es responsable del característico color rojo de la sangre [15].

3.2.2. Plaquetas

Las plaquetas, también llamadas trombocitos, son fragmentos de células unidas por membranas derivadas de la fragmentación de células precursoras más grandes llamadas megacariocitos, que provienen de la derivación de células madre de la médula ósea. Las plaquetas son fundamentales para el proceso de coagulación de la sangre, por lo que resultan esenciales para la cicatrización de las heridas. Debido a que carecen de núcleo, no contienen ADN nuclear [16].

3.2.3. Leucocitos o glóbulos blancos

Son componentes fundamentales para la defensa del organismo ante enfermedades y agentes externos. Además, eliminan células con ADN mutado y limpian los deshechos.

Los leucocitos abandonan continuamente el torrente sanguíneo para realizar sus funciones defensivas en los tejidos del cuerpo. Para los GB, la red vascular es simplemente un medio por el que viajar y eventualmente abandonar para llegar a su verdadero destino. Una vez que han salido de los capilares, algunos leucocitos toman posiciones fijas en el tejido linfático, la médula ósea, el bazo, el timo u otros órganos [cap. 18.4, 14].

3.2.3.1. Tipos de Leucocitos

Los GB se agrupan en tres clases principales -**linfocitos, granulocitos y monocitos**- según su morfología bajo la observación microscópica.

- Los **linfocitos** son responsables del reconocimiento específico de agentes extraños y de su posterior eliminación del organismo.
    - Las **células NK (*Natural Killer*)** responden a células parasitadas, tumorosas y regulan la respuesta inmunitaria. Son menos selectivos que otros linfocitos; ergo reconocerán y atacarán cualquier antígeno extraño en cualquier membrana [17].
    - Las **células T citotóxicas** (células TC) coordinan mecanismos de inmunidad migrando a focos de invasión de células foráneas y atacando células en conjunto con otros linfocitos. Las **células T colaboradoras** (células TH) activan las células B, que a su vez producen anticuerpos. Las células **T supresoras** (células TS) inhiben la activación de las células T y B, moderando la respuesta inmunitaria.
    - Las **células B** son responsables de la inmunidad mediante el uso de anticuerpos. Las **células B activadas** se diferencian en células plasmáticas, que fabrican anticuerpos en respuesta a antígenos extraños. A diferencia de las células T, las células B no tienen que migrar al lugar de la invasión, ya que los anticuerpos que sintetizan pueden viajar por toda la circulación [18].
- Los **granulocitos**, los más numerosos de los glóbulos blancos, liberan el cuerpo de grandes organismos patógenos como los protozoos o los helmintos y son también combatientes clave de la alergia y otras formas de inflamación [17].
    - **Neutrófilos:** Forma más común de leucocitos, representando el 50-70% de los glóbulos blancos circulantes. Reconocen las bacterias que han sido marcadas con anticuerpos o proteínas.
    - **Basófilos:** Los basófilos son menos comunes, representan el 1% de los leucocitos y son más pequeños que los neutrófilos. Su principal manera de proceder es la migración a los focos de lesión y la subsiguiente liberación de histamina y heparina. La histamina causa vasodilatación y la heparina previene la coagulación de la sangre, asegurando que se mantenga el suministro de sangre a la herida.

- **Eosinófilos:** Representan el 2-4% de los leucocitos circulantes. Son similares en tamaño a los neutrófilos, también son reconocidos por su núcleo bilobulado y sus abundantes gránulos rojos. Fagocitan las bacterias marcadas con anticuerpos y los restos celulares, pero su principal método de ataque a los cuerpos extraños es la liberación de compuestos tóxicos. Los eosinófilos también responden a los alérgenos y otra función es reducir la inflamación, restringiendo las acciones inflamatorias de los neutrófilos y los mastocitos en el foco de la lesión o infección [18].

- Los **monocitos**, que constituyen entre el 4 y el 8 % del número total de glóbulos blancos en la sangre, migran de ésta a los focos de infección, donde se diferencian en macrófagos, fagocitando así a microorganismos y restos celulares [17].

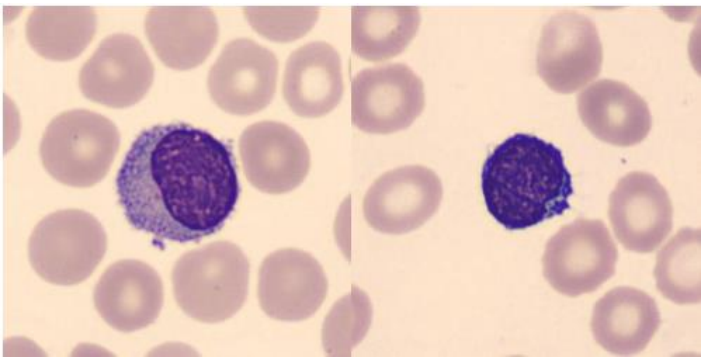


Figura 3.1. Ejemplos de linfocitos normales. En el de la izquierda se diferencia claramente el citoplasma (azulado claro que rodea el núcleo) del núcleo (morado oscuro). En la célula de la derecha se ve un núcleo con la cromatina más madura y apenas se aprecia el citoplasma. Las células de alrededor son eritrocitos. Fuente: dataset del Hospital Clínic.

3.3. Sistema linfático

El sistema linfático es una vía complementaria por medio de la cual sustancias líquidas pueden fluir desde los espacios intersticiales a la sangre. Es gracias a éste que se producen intercambios de proteínas y macropartículas entre espacios tisulares y capilares sanguíneos.

Este mecanismo de intercambio de fluidos es de vital importancia para filtrar el líquido de los extremos arteriales de los capilares sanguíneos que fluyen entre las células y no entra por los extremos venosos de otros capilares sanguíneos (1/10 del total). Por ende, este líquido entrará por los capilares linfáticos y volverá a la sangre por el sistema linfático mediante la linfa. Este proceso es necesario por el alto peso molecular de, por ejemplo, las proteínas que impide la perfusión a los tejidos. Los capilares linfáticos facilitan este proceso gracias a su estructura especial [19].

El sistema linfático se divide comúnmente en los órganos linfoides primarios, que son los lugares de maduración de las células B y T, y los órganos linfoides secundarios, donde deviene una más abundante

maduración de los linfocitos. Los órganos linfoides primarios incluyen el timo, la médula ósea y el hígado.

Los órganos linfoides secundarios incluyen los ganglios linfáticos, el bazo, pequeñas masas de tejido linfático como los parches de Peyer[1], el apéndice y las amígdalas. También constituyen el tejido linfoide asociado a las mucosas referido en la literatura como folículo linfoide (situado en tubo digestivo, bronquios, nariz, etc.). Este folículo está formado por una zona periférica o manto, constituido por linfocitos pequeños de cromatina madura, un centro germinal y una zona más externa llamada marginal.

Los órganos linfoides secundarios cumplen dos funciones básicas: dar lugar a la maduración de los linfocitos y atrapar eficazmente los antígenos [20].

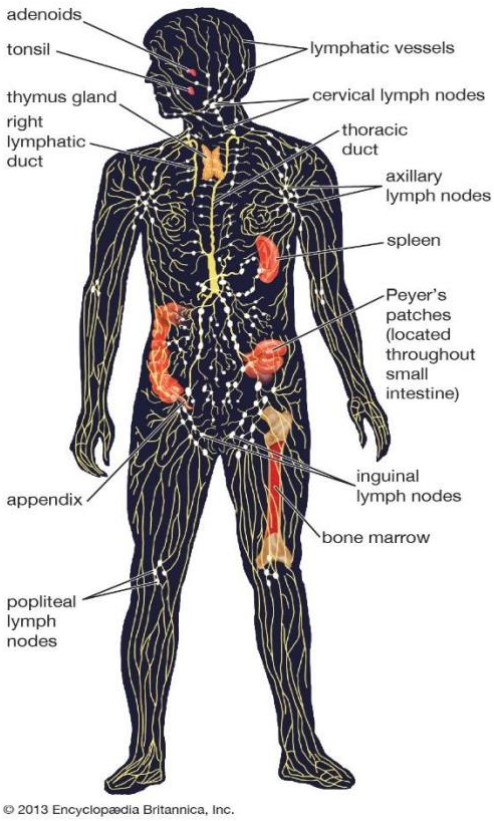


Figura 3.2. Mapa del sistema linfático. Fuente: [20].

3.4. Hematopoyesis

La hematopoyesis (formación de componentes celulares sanguíneos) ocurre durante el desarrollo embrionario y la edad adulta para generar y reponer el sistema sanguíneo [21].

La **eritropoyesis** es el desarrollo de glóbulos rojos. A medida que sus células precursoras maduran, su tamaño disminuye, la cromatina se condensa, los núcleos se extruyen y el citoplasma cambia de color de azul a rosa [22]. El proceso análogo en el caso de las plaquetas se llama **trombopoyesis**, esta vez pasando por un antecesor en concreto, el megacariocito [23].

El proceso de generación de leucocitos se realiza a partir de las células madre hematopoyéticas pluripotentes de la médula ósea. Existen dos vías importantes para generar varios tipos de leucocitos:

[1] Parches de Peyer: Nódulos de las células linfáticas que forman parches y que generalmente se presentan sólo en la porción más baja del intestino delgado.

Mielopoyesis, en la que los leucocitos de la sangre derivan de las células madre mieloides (sistema inmunitario innato), y **linfopoyesis**, en la que los leucocitos del sistema linfático (linfocitos) son generados a partir de las células madre linfoides (s. inmunitario específico) [24].

- **Linfopoyesis**: Las comentadas células madre linfoides multipotentes se diferencian en linfocitos T (cuya maduración tiene lugar en el timo), linfocitos B (maduración en la médula ósea), células de plasma y NK (maduración en médula ósea, ganglios linfáticos, bazo, amígdalas y timo).
- **Mielopoyesis**: A partir de células madre mieloides multipotentes se da lugar a mielocitos (estado intermedio que derivará a granulocitos y monocitos). Este proceso también produce células precursoras de macrófagos y células dendríticas que se encuentran en el tejido linfoide. Las células mieloides comparten un progenitor común conocido como la unidad formadora de colonias de granulocitos (CFU-GM). Esta célula surge de progenitores anteriores, que son capaces de dar lugar a linajes mieloides, megacariocíticos y linfoides, y en última instancia de la célula madre multipotente, que es capaz de diferenciarse en todas las líneas de células hematopoyéticas (ver Figura 3.3). La progenie de la CFU-GM es capaz de diferenciarse hacia granulocitos o monocitos y macrófagos [25].

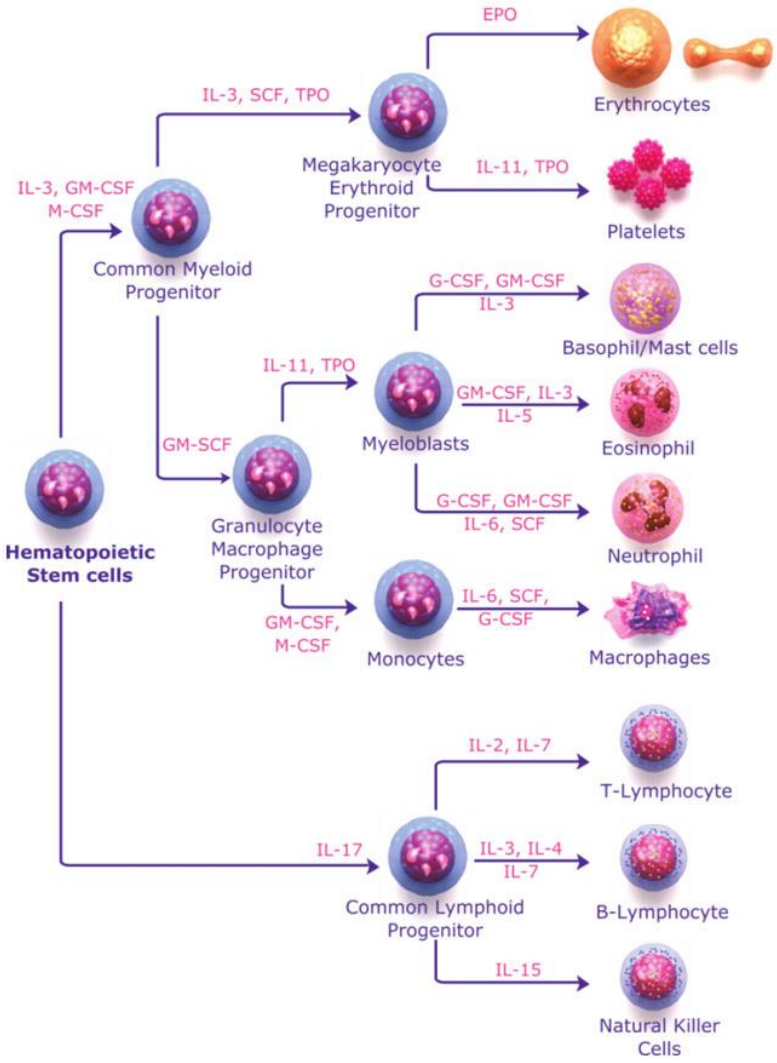


Figura 3.3. Hematopoyesis: distínganse de arriba abajo: eritropoyesis, trombopoyesis, mielopoyesis y linfopoyesis. Fuente: [26].

3.5. Anomalías en leucocitos

Las anomalías de los leucocitos más representativas del *dataset* de entrenamiento de las RNs de este trabajo serán las neoplasias linfoides (proliferaciones clonales de linfocitos en sus etapas de diferenciación, ocasionadas por alteraciones del sistema de inmunidad o agentes infecciosos) [p. 196, 27]. Se enumeran a continuación las entidades que se han seleccionado: Linfoma de la zona marginal esplénico (LZME), linfoma de células del manto (MCL, *Mantle Cell Lymphoma*), linfoma folicular (FL, *Follicular Lymphoma*) y la leucemia linfocítica crónica (LLC). Los blastos linfoides o células inmaduras pertenecientes a leucemias linfocíticas agudas también merecen una especial mención ya que también se clasificarán. Además, se incluirá un grupo de linfocitos reactivos presentes en la sangre en enfermedades infecciosas no malignas, ya que pueden ser confundidos con las anteriores malignidades y también forman parte del *dataset*.

- **Linfoma de la zona marginal esplénico**: Anomalía indolente y poco frecuente. Causa una proliferación autoreplicativa descontrolada de linfocitos B maduros. Compromete la médula ósea y bazo y provoca frecuente leucemización en sangre. Mayormente, presenta agrandamiento patológico, aumento de la producción de linfocitos, disminución de la proporción de GRs y plaquetas. Puede estar ocasionado por la contracción hepatitis C y algunas patologías autoinmunes [28].

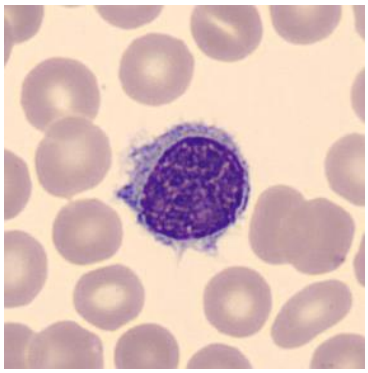


Figura 3.4. Célula del linfoma de LZME. Fuente: dataset del Hospital Clínic.

- **Linfoma folicular**: Proliferación de células B cuya estructura nodular de la arquitectura folicular se conserva. Se encuentra principalmente en los ganglios linfáticos, pero también puede afectar a bazo, médula ósea, sangre periférica y anillo de Waldeyer (conjunto de estructuras de tejido linfoide localizadas en la faringe). Tiene lugar en forma de alteraciones mutacionales en linfocitos B de pequeño tamaño, escaso citoplasma y perfil irregular de núcleo cuando estos se leucemizan (llegan a la sangre periférica) [29] [30] [pp. 206-207, 27].

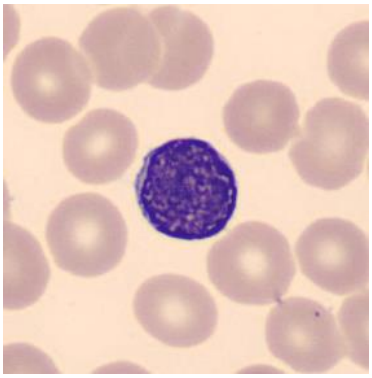


Figura 3.5. Célula del Linfoma Folicular. Fuente: dataset del Hospital Clínic.

- **Linfoma de células del manto**: El linfoma de células del manto (LCM) se desarrolla cuando se produce una proliferación anómala de células B maduras, las células características de este tipo de linfoma [31]. La médula ósea se ve afectada precozmente y las células linfoides anormales muestran un tamaño variable entre mediano y pequeño además de polimorfismo. Se reconocen varios tipos, pero se destacarán los tres que están contenidos en el *dataset*: blástica

(alto nivel proliferativo, cromatina laxa e inmadura), el indolente y la variedad típica o clásica [pp. 207-208, 27].

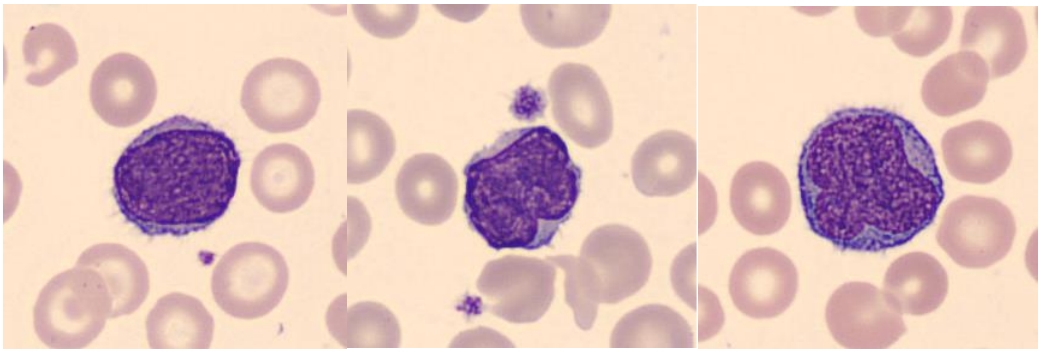


Figura 3.6. De izquierda a derecha: LCM Blástico, LCM Indolente, LCM Típico. Fuente: dataset del Hospital Clínic.

- **Leucemia linfática crónica (LLC):** Linfocitos que prolongan su vida anormalmente al postergar su muerte programada (apoptosis) por lo que aumenta lentamente su número observándose una proliferación en los recuentos sanguíneos y en la médula ósea. Por consiguiente, se debilita enormemente el sistema inmunitario [32]. Las células linfoides anormales se presentan con cromatina madura de aspecto "*grumelee*". La ratio núcleo/citoplasma es elevado. Al ser frágiles, pueden romperse en el frotis generando sombras nucleares de Grumpecht [p. 197-199, 27] como se muestra en la Figura 3.7.

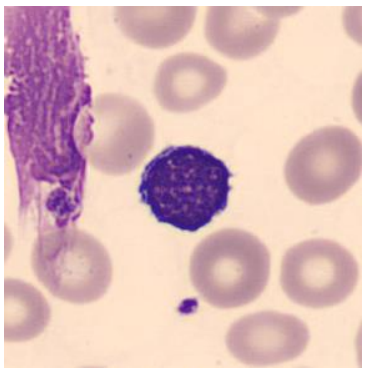


Figura 3.7. Linfocito anormal de la leucemia linfocítica crónica, nótese la sombra de Grumpecht de otro Linfocito anormal roto. Fuente: dataset del Hospital Clínic.

- **Linfocitos reactivos o variantes:** Linfocitos B, T o NK grandes, de atípica apariencia y que se asocian a infecciones, siendo la más frecuente la mononucleosis. El aspecto de la cromatina tiende a ser difuso o parcialmente condensado, y el citoplasma basofílico (azulado). Pueden confundirse con blastos [33].

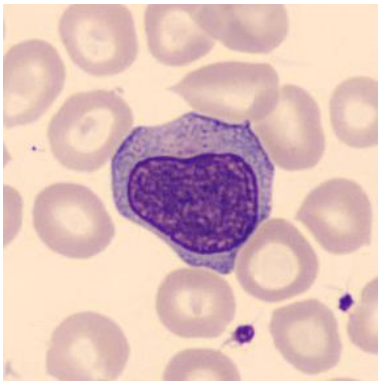


Figura 3.8. Linfocito Variante, también llamado Reactivo. Fuente: dataset del Hospital Clínic.

- **Blastos:** Se entienden por esta nomenclatura a los leucocitos inmaduros leucemizados en sangre periférica proliferantes en leucemias agudas: linfoides y mieloides. La morfología cambia de forma distintiva en los subgrupos de ambas patologías.

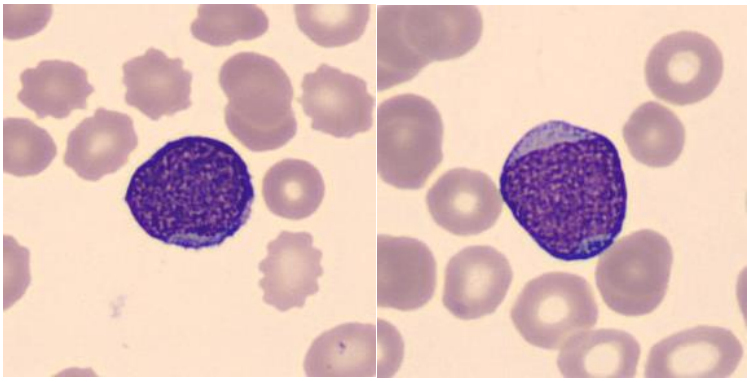


Figura 3.9. (Izquierda) Blasto de LAL-T, Leucemia aguda linfoide, un linfocito T. (Derecha) Blasto de LAM, Leucemia Aguda Mieloide. Fuente: dataset del Hospital Clínic.

3.6. Observación sanguínea: Frotis

Las técnicas citológicas y concretamente hematológicas son de vital importancia para este trabajo, ya que constituyen el mecanismo de obtención de imagen celular para el estudio de leucocitos, pudiendo caracterizar enfermedades. La biología celular está sujeta al desarrollo de instrumentación y tecnologías ya que los estudios son sensibles a la calidad de estos dispositivos. Se destaca la prueba de frotis de SP como la observación microscópica de una extensión de este fluido corporal sobre un portaobjetos. Es crucial para el diagnóstico y la monitorización de la progresión de la enfermedad y la respuesta terapéutica, ya que una comprensión experta de la interpretación de la SP es importante para una práctica clínica exitosa [34].

Comúnmente, la sangre se obtiene de las venas periféricas o de extracciones digitales y se almacena en un frasco anticoagulante, preferiblemente *EDTA*[2], para la decantación de ésta generando separación entre plasma, células nucleadas con plaquetas y hematíes. Posteriormente se realiza una extensión, colocando una gota de sangre sobre un portaobjetos (fina hoja de material transparente y rectangular) y se arrastra con otro formando un ángulo de 45°. A continuación, se procede a la fijación por metanol (para inmovilizar los tejidos celulares) y a la tinción (comúnmente se utiliza el procedimiento con oxidantes *May Grünwald-Giemsa*) [27].

Cuando tiene lugar la observación microscópica, ya sea manual o automatizada, se procede a realizar la evaluación morfológica de los elementos sanguíneos y el recuento diferencial de los distintos tipos de leucocitos. Previamente se realiza la determinación de parámetros hematológicos básicos que nos informan de posibles alteraciones cuantitativas de los eritrocitos, leucocitos o plaquetas [p. 1, 27].

Para ello, la manera de proceder suele ser la introducción de la muestra de sangre en un autoanalizador, donde se cuantifica, clasifica y dibuja una distribución de los diferentes tipos de células, por medio de técnicas electrónicas y ópticas. Acto seguido se utiliza un *auto-stainer* para aplicar la tinción y obtener la extensión. Finalmente se introduce la extensión en un equipo motorizado como el modelo DM96 y se obtienen las imágenes, con posibilidad de ser preprocesadas, normalizadas y/o preclasificadas tipológicamente [p. 16, 4].

Cabe destacar que los modelos DM de la firma Cellavision explicitan el número de extensiones cargables en su modelo, teniendo el *DM96* la capacidad de cargar 96 *slides* y el modelo DM1200 12 *slides* [p. 4, 1].

[2] Ácido etilendiaminotetraacético

4. Bases del Machine Learning, Deep Learning y GANs

4.1. Machine Learning

Tradicionalmente, se conoce como Machine Learning (ML) al campo de estudio o rama de la inteligencia artificial que utiliza distintas técnicas (tanto matemático-estadísticas como computacionales) con el fin de que un algoritmo o máquina sea capaz de aprender a resolver un problema. Cabe destacar que estos algoritmos o **modelos** aprenden de un conjunto de datos que contienen registros, observaciones, imágenes, ejemplos o muestras de alguna índole en concreto. Mediante la **experiencia** o **entrenamiento**, se busca que el modelo pueda predecir una variable, clasificar una muestra o realizar algún tipo de inferencia sobre otros datos de la población o parte de esta [35].

Los algoritmos de ML destacables en lo que respecta a este TFE son los siguientes.

- **Aprendizaje supervisado**: Donde el algoritmo utiliza una función de aprendizaje donde a una entrada o *input* se le asigna una salida u output (en la práctica una etiqueta), por lo que se dice que el conjunto de datos de entrenamiento está basado en ejemplos pares entrada-salida. [36]. Existen numerosas técnicas de *supervised learning*, entre ellas destacan **regresión lineal**, **regresión logística**, **análisis de discriminante lineal**, **máquinas de soporte vectorial**, **árboles de decisión**, **k-vecinos cercanos** (k-nearest neighbours), **redes neuronales**, etc.

- **Aprendizaje no supervisado:** Es un tipo de aprendizaje auto-organizado que ayuda a encontrar patrones previamente desconocidos en el conjunto de datos sin etiquetas preexistentes [37]. Dos de los principales métodos utilizados en el aprendizaje no supervisado son el análisis de componente principal (**PCA**) [38] y el ***clustering*** [39].

4.2. Fundamentos de Redes Neuronales

4.2.1. Multilayer Perceptron y la Neurona Artificial

Según Foster, D. [p. 31, 40], el aprendizaje profundo es una clase de algoritmo de aprendizaje autónomo que utiliza múltiples capas apiladas de unidades de procesamiento para aprender representaciones de alto nivel a partir de datos no estructurados. Estas unidades de procesamiento son conocidas formalmente como capas de neuronas artificiales (en el caso del **MLP**, ***Multilayer Perceptron***). Existen

diferentes tipos de capas según la arquitectura de la RN, si éstas están constituidas por **filtros** (también llamados ***kernels)*** se tratará de una CNN, **red neuronal convolucional**.

Para entender el MLP, es necesario comprender el funcionamiento de una **neurona artificial** o **perceptrón,** concepto introducido por Frank Rosenblatt en 1958, basado en el modelado del comportamiento de la neurona biológica transmitiendo sus potenciales de acción [41]. Como se puede intuir en la Figura 4.1, una unidad neuronal recibe unos inputs en forma de constante numérica [x_0, x_n], estos son multiplicados por sus respectivos **pesos** (***weights***) y sumados, añadiendo a este sumatorio un coeficiente numérico llamado ***bias*** (en español **sesgo**). Después de esta suma ponderada, el resultado pasará por una **función de activación** (también llamada no linealidad), en este caso una función escalón. Dicho procedimiento quedaría expresado de la siguiente manera:

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \qquad \textbf{(Ec. 1)}$$

En la Ec. 1, *w* representa el vector de pesos matriz de pesos [w_0, w_1, …, w_{n-1}, w_n], mientras que *x* representa el vector de entradas de la neurona [x_0, x_1, …, x_{n-1}, x_n] y *b* a su vez el *bias*. Por ende, el producto *w·x,* simboliza el sumatorio ponderado $\sum_{i=0}^{n} wi \cdot xi$. Si este sumatorio es superior al *bias,* la salida de la neurona será 1, es decir, se habrá activado o encendido. De lo contrario, la salida será 0, pero esto es debido a que la función activación es una función escalón. Posteriormente se explicarán otras funciones de activación que modificarían el output de la neurona (resultado numérico) de diferente manera.

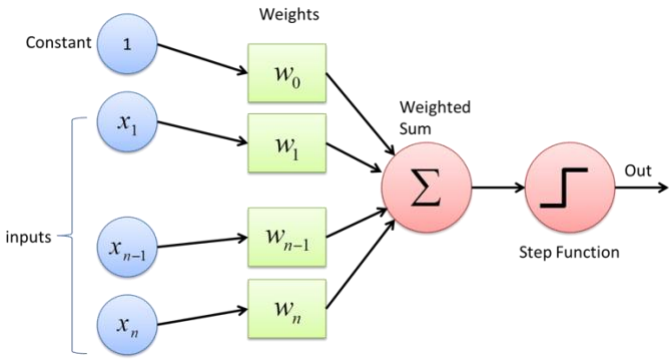


Figura 4.1. Diagrama del perceptrón o neurona artificial. Fuente: [42].

En la literatura, las salidas de cada neurona son llamadas también **activaciones** y servirán de entrada para las neuronas de capas siguientes, multiplicadas por otros pesos *w*, y así sucesivamente hasta las últimas capas constituyendo el MLP, véase la Figura 4.2. En general, los coeficientes ***w*** y ***b*** se conocen con el nombre de **parámetros,** mientras que las salidas de cada neurona, como ya se ha comentado, activaciones [p. 33, 46]. La entrada de la capa de *input* puede ser una imagen, espectrograma de sonido,

datos tabulados, texto, etc. Las capas que no son *input* ni *output* se llaman ocultas y realizan la computación numérica responsable del aprendizaje y predicción.

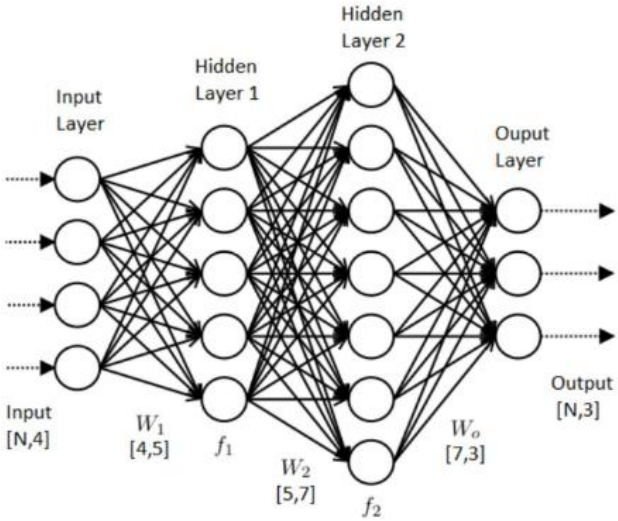


Figura 4.2. Representación gráfica de un MLP. Los círculos representan las activaciones, las flechas los pesos, y en los números entre corchetes, la cantidad de activaciones de la capa anterior y posterior. Fuente: [41].

4.2.2. Funciones de activación

La adición funciones de activación (también llamadas no linealidades) al algoritmo responde a la limitación de los modelos lineales para solucionar problemas linealmente no separables por un hiperplano, siendo el ejemplo más famoso el de la función XOR [p-35, 43]. Como regulando los parámetros de una función lineal no se puede encontrar solución a este problema, se aprovecha el potencial de la función de activación del perceptrón. La idea de poder aproximar cualquier función matemática mediante combinaciones de funciones no lineales se basa en el Teorema de Aproximación Universal [44].

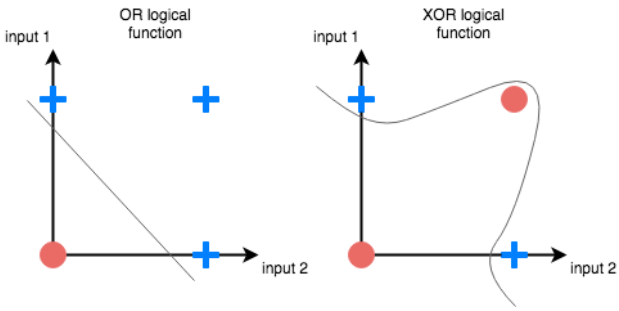


Figura 4.1. XOR Problema de las puertas lógicas XOR y su solución. Fuente: [45].

En alto nivel, estas funciones permiten que las capas de la RN puedan aprender y detectar características de los datos, cuanto más profunda es la capa, mayor la complejidad de los patrones

reconocidos. Además, éstas impiden que la RN colapse en una sola función, ya que, si todas las funciones del modelo fueran lineales, se reduciría en sola una función lineal compuesta por todas [46].

Entre las funciones de activación más utilizadas destacan las que se muestran en la Figura 4.4:

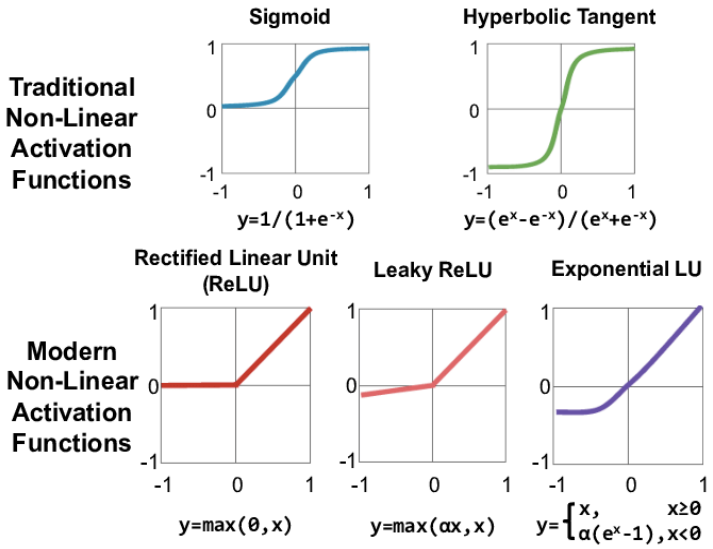


Figura 4.4. No linealidades. Fuente: [47].

- **Sigmoide:** Su salida está contenida en el intervalo [0, 1]. Los valores de la salida tienden a 1 o 0 porque por efecto de pequeños cambios del dominio de *X,* se esperan grandes cambios en *Y* en valores muy cercanos a los límites del rango. Es por eso que se dice que esta función satura, además de generar gradientes muy pequeños [49].
- **Tanh:** La tangente hiperbólica es parecida a una sigmoide re-escalada al intervalo [-1,1] en el dominio *Y*. Cabe destacar que el gradiente es más fuerte para la función tanh que para la sigmoide (las derivadas son más pronunciadas) [49].
- **ReLU (*Rectified Linear Unit*):** La unidad de rectificación lineal convierte la entrada en cero si es negativa y a mantiene si es positiva. Según *Krizhevsky et al. (2012)* [48] aumenta en un factor de 6 la rapidez de aprendizaje respecto a las dos anteriores. Favorece en el entrenamiento que en una entrada normalizada se reduzca al aproximadamente 50% el número de neuronas en funcionamiento. No obstante, esto puede acarrear la problemática del *Dying ReLU* [49].
- **Leaky ReLU:** Para mitigar el problema mencionado anteriormente, se modifica la ReLU para crear la Leaky ReLU, sustituyendo la línea horizontal del 0 en una recta ligeramente inclinada. Esto pretende no anular los gradientes permitiendo reajustar los parámetros de la red con más asiduidad [49].
- **ELU:** La unidad exponencial lineal ha demostrado mejorar los desempeños de las RN en reconocimiento de imágenes en *Clevert et al. (2015)* [50] frente a otras funciones de activación. Sustituye la recta de la Leaky ReLU por una exponencial.

- **Softmax:** Se ha querido hacer una mención especial a esta función, comúnmente utilizada en la capa de output de RNs de clasificación múltiple de un número *j* de categorías. Véase representada en la Figura 4.5.

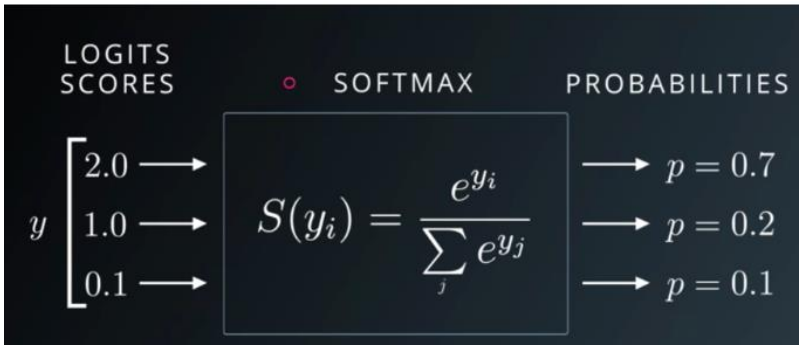


*Figura 4.5. Los **logits** o activaciones de la última capa pasan por dicha función generando un vector de probabilidades correspondientes a cada categoría, un número real entre 0 y 1. Fuente: [51].*

4.2.3. Función de Coste o Pérdida

Tomando como salida las activaciones de la última capa de la RN habiendo pasado por la función *softmax* en un problema de clasificación multi-clase, existe la necesidad de comparar las probabilidades predichas de la salida con la ***ground truth*** (categoría real).

Para ello, será necesario configurar una función de coste. Ésta cuantificará el desempeño de la red neuronal a la hora de realizar sus predicciones y servirá como dirección a la que avanzar para aprender de los ejemplos y del propio error. La función de coste reduce todos los diversos aspectos buenos y malos de un sistema posiblemente complejo a un único número, un valor escalar que permite clasificar y comparar las soluciones posibles [pp. 155-156, 52].

Es un ejemplo el error cuadrático medio (***MSE, Mean Squared Error***) popular para los problemas de regresión (*output/s* de variable numérica real). También está la función de entropía cruzada (***cross-entropy error***), que se utiliza a menudo para problemas de clasificación cuando los resultados se interpretan como probabilidades de pertenencia a una clase indicada (ya sea binaria −dos categorías− o multi-clase, más de dos). Véase Ec. 4 [52].

En la Ec. 2 se muestra el *MSE* o *Quadratic Loss* (*L2)*. Se promedia la diferencia cuadrada entre las predicciones "*ŷ*" las observaciones reales *"y"*. Esto facilita el cálculo de gradientes [54].

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \quad \textbf{(Ec. 2)}$$

En la Ec. 3 se muestra el error absoluto medio (*MAE*) o *L1*. Similar al MSE, promedia la suma de las diferencias absolutas entre las predicciones y las observaciones reales [54].

$$MAE = \frac{\sum_{i=1}^{n} \mid y_i - \hat{y}_i \mid}{n} \quad \textbf{(Ec. 3)}$$

A la hora de comparar una imagen transformada o creada por una GAN con una imagen objetivo suelen utilizarse las funciones de pérdida *L1* o *L2* para ver cuán errónea es la transformación (a nivel de similitud de píxeles calculada por estas funciones).

La siguiente función de pérdida es la **cross-entropy loss** (Ec. 4)**.** Para una implementación de esta función en una clasificación multi-clase se aprovechan las salidas de la función *softmax* y un vector de probabilidades (elementos $f(s)_i$), tal y como se muestra en la Figura 4.6. [55].

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)) \quad \textbf{(Ec. 4)}$$

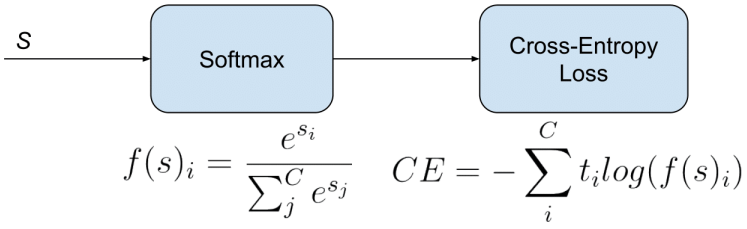


Figura 4.6. Diagrama del proceso de cálculo de la cross-entropy loss en una clasificación multi-clase. Esta función de pérdida aumenta cuanto más diverja la probabilidad de la clase real de la predicción hacia las otras clases. Fuente: [55].

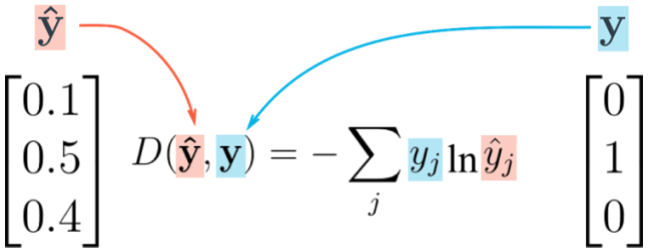


Figura 4.7. Diagrama de los vectores de probabilidad softmax (izquierda) y el vector one-hot (derecha). Como se ve, solo se conservará el 0.5 en el sumatorio al multiplicar el logaritmo de las probabilidades por un vector 1-hot-encoded (derecha). Fuente: [56].

4.2.4. *Backpropagation* y *SDG* como Algoritmos de Optimización

El *Backpropagation* es el algoritmo de aprendizaje de las RN popularizado por *Rumelhart et al.* (1986) [57] en su artículo titulado *"Learning representations by back-propagating errors"*. Según *Rumelhart et al.* (1986), el procedimiento ajusta repetidamente los pesos (y *bias*) de las conexiones en la red para minimizar la función de pérdida. Este ajuste de parámetros se realiza cada vez que uno o varios *inputs* entran por la RN y se propagan hacia el *output* para que se extraigan los *logits.* Por consiguiente, se calculan la función de coste y el posterior gradiente, es decir un ***feedforward pass.*** Una vez esto tiene lugar, se aplica la regla de la cadena para propagar la actualización correspondiente a cada parámetro. La regla de la cadena se utiliza para sustraer parte del gradiente a cada parámetro teniendo en cuenta su contribución a la variación del propio gradiente de la función de coste.

Se muestra en las Ec. 5 y 6, la regla de la cadena aplicada a una RN. Siendo *C* la función coste, R la función de activación, Z el producto de pesos e input, *X* y *W* las matrices de input y parámetros respectivamente, se calcularía $C'(W_i)$ aplicando esta regla para cada W_i.

$$Cost = C(R(Z(XW))) \quad \textbf{(Ec. 5)}$$

$$C'(W) = C'(R) \cdot R'(Z) \cdot Z'(W) = (\hat{y} - y) \cdot R'(Z) \cdot X \quad \textbf{(Ec. 6)}$$

Para ello es necesaria la aplicación del **SDG** (***Stochastic Gradient Descent)***, que consistente en utilizar el gradiente de la función de coste en sentido negativo repetidas veces con varios ejemplos del *dataset* de entrenamiento a la vez (**mini-batchs**, de donde se extrae la media del gradiente que generan los ejemplos que contienen). Matemáticamente, el gradiente se define como el vector fruto de la diferenciación de la función respecto a todas las variables [58]. Además, señala la dirección en el espacio multidimensional hacia la pendiente máxima en sentido ascendente (sentido positivo) y descendente (sentido negativo). Intuitivamente, el mínimo de la función se situará en sentido descendente del gradiente en un punto aleatorio de la representación espacial de la función de pérdida, véase la Figura 4.8.

En la Ec. 7 se muestra la definición de gradiente definida para una función *f* de *n* variables reales. Queda entonces el gradiente expresado como un campo vectorial cuyos componentes son las derivadas parciales de f evaluados en un punto *p* [58].

$$f\colon \mathbf{R}^n \to \mathbf{R} \qquad \nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix} \quad \textbf{(Ec. 7)}$$

Para que el modelo converja de forma rápida y estable es necesario actualizar con *mini-batches* por razones de arquitectura y optimización en la GPU [59].

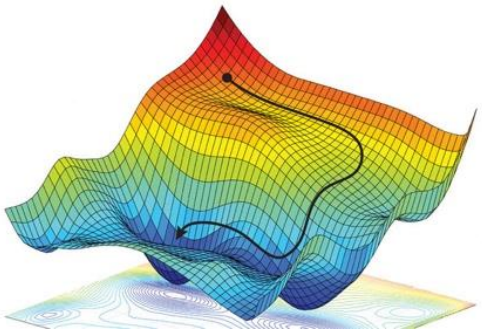


Figura 4.8. Representación gráfica del SDG en una función. Fuente: [60].

En la Ec. 8 se muestra un ejemplo de paso realizado para actualizar el conjunto de parámetros Θ. Se le substrae a Θ una porción llamada *learning rate* (α) del gradiente ∇J. Esta función (gradiente) depende de las predicciones del *mini-batch x(i)* y de sus etiquetas*, y(i)*.

$$\theta = \theta - \alpha \frac{1}{n} \sum_i^n \nabla J\left(\theta; x(i), y(i)\right) \quad \textbf{(Ec. 8)}$$

Volviendo a la Figura 4.8, cabe destacar que la substracción que se le realiza al conjunto de parámetros de la RN no es de todo el gradiente, sino de un porcentaje de este para obtener una convergencia más estable. Es la llamada ***Learning Rate***, $\boldsymbol{\alpha} \in (0,1]$ (velocidad a la que aprenden las capas de la red) [46]. *Fastai* propone la utilización de *learning rates* discriminativas, es decir, que las primeras capas aprendan a una menor velocidad que las últimas ya que interesa retener información de las estructuras aprendidas por las primeras capas en modelos entrenados [46].

4.2.4.1. Estrategias de optimización adaptativas, mejoras del SDG

Son métodos diseñados para acelerar el aprendizaje variando la *learning rate* según devenga el entrenamiento (regularidad o irregularidad de gradientes, grandes curvaturas del espacio de la función de coste, etc). Destacan el método del ***Momentum*** [61], ***RMSProp*** [62] y ***Adam*** [63]. Los dos últimos varían la *learning rate* para cada parámetro personalizadamente.

En la Ec. 9 se muestra el algoritmo del *Momentum*. "*f*" es la predicción del modelo mientras que "*L*" es la función de coste. Dentro de "*L*" *se* computa la pérdida asociada a las predicciones "*f*" con respecto a las etiquetas "*y*". Siendo "ε" el coeficiente de *Momentum*, "*m*" el tamaño del *mini-batch,* "θ" el conjunto de parámetros*,* "α" el learning rate y "*v*" la acumulación de velocidades, se substrae al producto "αv" de la iteración anterior el producto "$\varepsilon \nabla \theta$" de la iteración actual. Posteriormente se añade al conjunto de parámetros θ la nueva "*v*" como iteración del conjunto actual [p. 311, 43].

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right)$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}. \qquad \textbf{(Ec. 9)}$$

4.2.5. Métricas, *underfitting*, *overfitting* y generalización

Las métricas son medidas para cuantificar la precisión o desempeño del modelo de RN que no tienen repercusiones en el aprendizaje. También podrían utilizarse como métricas las funciones de error comentadas anteriormente. En modelos generativos o de clasificación, podría considerarse la comprobación cualitativa de la imagen generada por parte de un experto la métrica para evaluar el desempeño de los modelos. Para ello se ha distinguir entre diferentes subconjuntos que se separan del *dataset:*

- **Conjunto de *training*:** Normalmente el 70% del conjunto de datos, se usa para entrenar el modelo.
- **Conjunto de *validation*:** Sirve para comprobar la eficacia o desempeño del modelo y para probar hiperparámetros como *learning rate*, número de capas, parámetros de optimización o regularización no entrenables que afectan al entrenamiento, etc. También parámetros de decisiones de diseño o arquitectura.
- **Conjunto de *testing*:** Puede ser opcional y no tiene implicaciones en el entrenamiento. Habitualmente se utiliza para evaluar la calidad de la clasificación con datos completamente ajenos al entrenamiento y a la validación-selección de parámetros. Es un paso previo a la puesta en producción del modelo una vez entrenado.

Las métricas ayudan a tomar consideraciones sobre la **generalización** que está llevando a cabo el modelo, es decir, qué comportamiento tiene el modelo frente a datos que no ha visto en el conjunto de entrenamiento. Esto significa que se puede saber si el modelo sufre ***underfitting*** u ***overfitting*** [64]. Este concepto es explicado también desde la excesiva complejidad del modelo (Figuras 4.9 y 4.10).

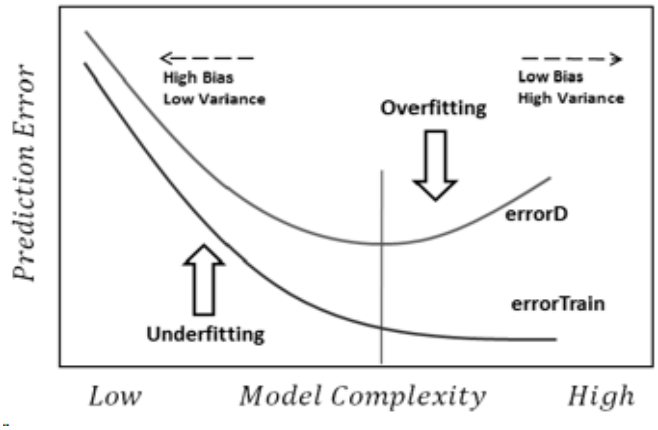


Figura 4.9. Evolución del error de predicción durante el entrenamiento (subiendo la complejidad), dando a entender que en el overfitting se dan resultados peores por sobre-entrenamiento. Fuente: [65].

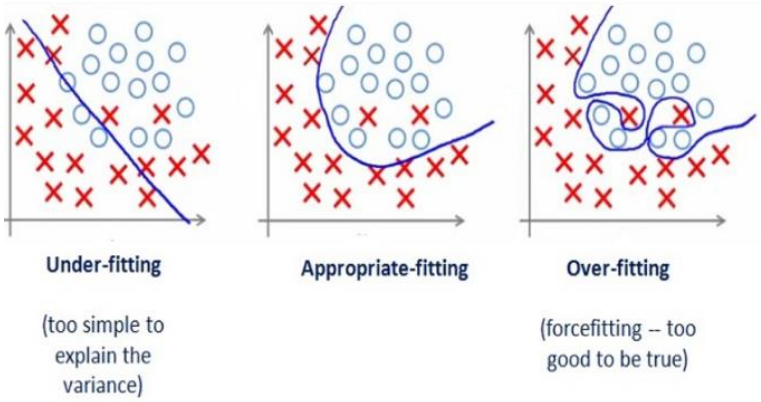


Figura 4.10. Efectos del overfitting y underfitting en un problema de clasificación, como se puede observar la solución del overfitting es demasiado compleja mientras que en el underfitting no explica la realidad de los datos. Fuente: [66].

Según J. Howard en sus ponencias en el curso de *Fastai* [64], en modelos sofisticados de DL de hoy en día como la ResNet es bastante difícil alcanzar *overfitting.* No obstante, el uso de técnicas de **regularización** para prevenirlo es muy habitual y se explicarán más adelante.

Para evaluar el modelo GAN de este trabajo se empleará la ***accuracy*** *(*también llamada desempeño del modelo*)* obtenida en base a la clasificación de las imágenes transformadas del Hospital Can Ruti con las CNN seleccionadas*.* Cuanto mayor sea esta, mejor se habrá logrado emular las características de las imágenes de Hospital Clínic. También se utilizará para medir el desempeño de los clasificadores creados para los leucocitos del Hospital Can Ruti.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{N\ predicciones\ Correctas}}{\boldsymbol{Npredicciones\ Totales}} \qquad \textbf{(Ec. 10)}$$

Se usará también la **matriz de confusión**, cuya diagonal muestra las cifras de las categorías predichas correctamente. Se distinguen en un eje las reales y en otro las predicciones.

También se utilizará la métrica de **precisión**. La precisión responde al total de elementos bien clasificados de una clase respecto al total de elementos clasificados como esa clase (ver Ec. 11, donde *Tp* representa *Total Positive* y *Fp* representa False *Positive*).

$$P = \frac{Tp}{Tp+Fp} \quad \textbf{(Ec. 11)}$$

4.2.6. Regularización

4.2.6.1. Weight Decay (decaimiento de pesos)

Esta técnica de regularización se basa en la adición de un componente en la función de coste que impide que el valor de los pesos se dispare, penalizando la complejidad del modelo tal como se expresa a continuación [43], [67]:

En la Ec. 12 se muestra el *Weight Decay* propiamente dicho, la adición del sumatorio de pesos multiplicado por el cociente del coeficiente λ entre el doble del número de *samples* que intervienen.

$$Coste = LossFunction + \frac{\lambda}{2m} * \sum||w|| \quad \textbf{(Ec. 12)}$$

Por el contrario, en la Ec. 13 se muestra la regularización *L2*, donde el sumatorio es de los pesos al cuadrado penaliza en mayor medida los pesos con valor muy alto.

$$Coste = LossFunction + \frac{\lambda}{2m} * \sum||w||^2 \quad \textbf{(Ec. 13)}$$

4.2.6.2. Dropout

Es una idea introducida por *Srivastava, N. et al. (2014)* [68], que propone el apagado o el acto de ignorar un conjunto de neuronas, tanto ocultas como visibles (capa *input*), con tal de simplificar el modelo a la hora de realizar el entrenamiento y que así no dependan de algunos nodos en concreto (también se conoce en la literatura como añadir ruido).

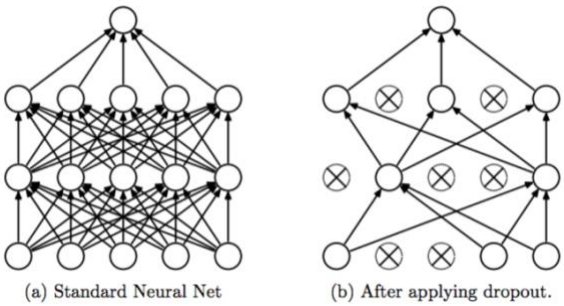


Figura 4.11. Diagrama del dropout: Fuente [68].

4.2.6.3. Data Augmentation

Es una técnica de pre-procesado y regularización demostrándose efectiva en estudios como el de *Jason Wang y Luis Perez* (2016) [69]. Se realiza aumentando el tamaño del conjunto del entrenamiento añadiendo copias adicionales de los ejemplos que han sido modificados con transformaciones que no cambian la clase (rotaciones, translaciones, granulaciones, cambios de orientación, adición de márgenes, etc.) [p.458, 43].

4.2.6.4. Batch Normalization

La normalización de lotes (BN) mejora tanto la convergencia como la generalización en el entrenamiento de redes neuronales [70]. Esta idea surge a raíz de querer escalar los *mini-batches* tal y como se hace en la entrada (restando la media y dividiendo entre la desviación estándar). Esta vez se realiza en las capas ocultas de la red neuronal y añadiendo al *mini-batch* un parámetro de escalamiento y otro de desplazamiento como en una recta *y = ax*. Estos dos parámetros, *γ* y *β* respectivamente, son ajustables durante el entrenamiento por medio de *Backpropagation*.

En la Ec. 14 se muestra la expresión del BN. "*g*" es la función de activación, "$\hat{h}$" y "*h*" son los valores ocultos antes y después de la BN, "*w*" es el vector de pesos y "*x*" es entrada numérica. "μB" y "σB" son la media y la desviación estándar de "*h*". Por último, "*γ*" y "*β*" son los parámetros de escala y desplazamiento del *mini-batch*.

$$y = g(\hat{h}),\ \ \hat{h} = \gamma\frac{h - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} + \beta \ \text{ and } \ h = \mathbf{w}^{\mathsf{T}}\mathbf{x}. \qquad \textbf{(Ec. 14)}$$

4.3. Redes Neuronales Convolucionales (Convolutional Neural Networks, CNN)

Las CNN son ampliamente utilizadas en tareas de visión por computador y su característica principal es que su operación fundamental realizada en las capas de la red es la convolución, en vez de un producto de matrices típico $w^T x + b$. La convolución consiste en pasar un *kernel* (también llamado máscara o filtro) de un tamaño NxN pixeles por la imagen realizando productos matriciales cuyas salidas originan los pixeles de otras imágenes resultantes, (llamadas **canales** o **activaciones)**. Estas imágenes serán el argumento de la posterior función de activación. Se destaca un parámetro llamado *stride* que cuantifica los saltos de pixeles que da el filtro por la imagen de entrada tras cada multiplicación [p. 11, 1].

Como se muestra en la Figura 4.14, la entrada de la red es un tensor de rango 3, ya que es una imagen de tres canales RGB. Por lo tanto, los *kernels* tendrán que ser a su vez tensores de rango 3. El caso es que normalmente entra más de una imagen en forma de *mini-batch* y por ende la entrada de imágenes será un tensor de rango 4 (*K* imágenes x *3* canales x *M* píxeles x *M* ó *N* píxeles). Los *kernels* también tendrán que ser tensores de rango 4 (K *Kernels* x 3 canales x M píxeles x *M* ó *N* píxeles). Nótese que la dimensión de *K kernels* está constituida por muchos filtros diferentes en cada capa de convolución. Los pesos de estos *kernels* se actualizan por *Backpropagation* [46].

Las capas de convolución se suelen intercalar con capas ***pool***, cuyo objetivo es el ***downsampling,*** y a veces la agrupación de activaciones. Esta capa no tiene pesos y simplemente disminuye la dimensión promediando o escogiendo los valores máximos de entre las entradas (en el mismo canal a nivel de píxeles o agrupando los canales si es ***average pooling***) [71]. Cuando se consigue una abstracción y reducción de dimensiones suficientes, se "aplana" el tensor de canales (o fusionando los canales si se utiliza el *avg pooling*) en un vector y este entra en una ***fully connected layer***, lo que sería un *MLP* de clasificación. Aquí, cada pixel de cada canal entraría por una entrada de neurona.

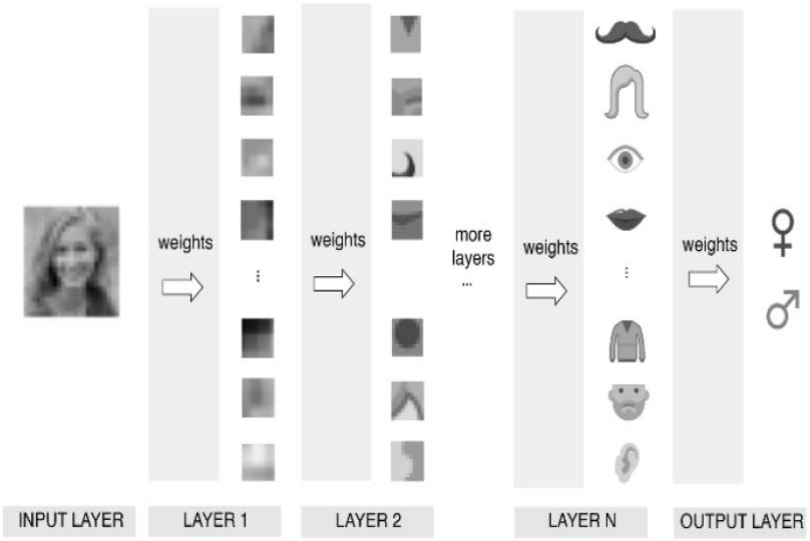


Figura 4.12. Diagrama de una red neuronal y un forwardstep para la detección de género de la persona de la imagen. Nótese que las características o estructuras detectadas tras cada capa son más complejas. Fuente: [40].

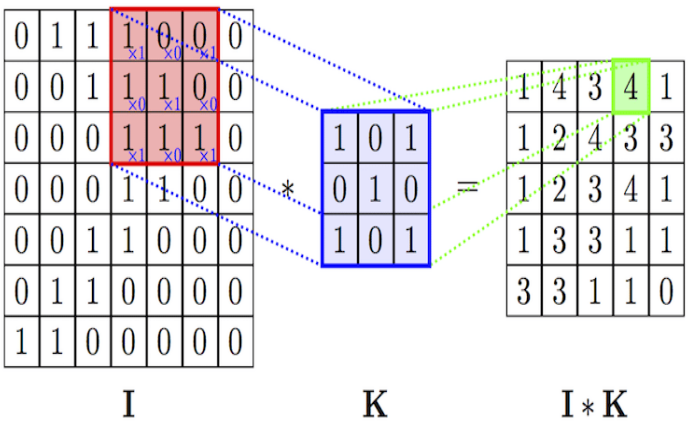


*Figura 4.13. Diagrama de una operación de convolución, nótese como deslizando el filtro K sobre la imagen I obtenemos otra imagen I*K. Fuente: [72].*

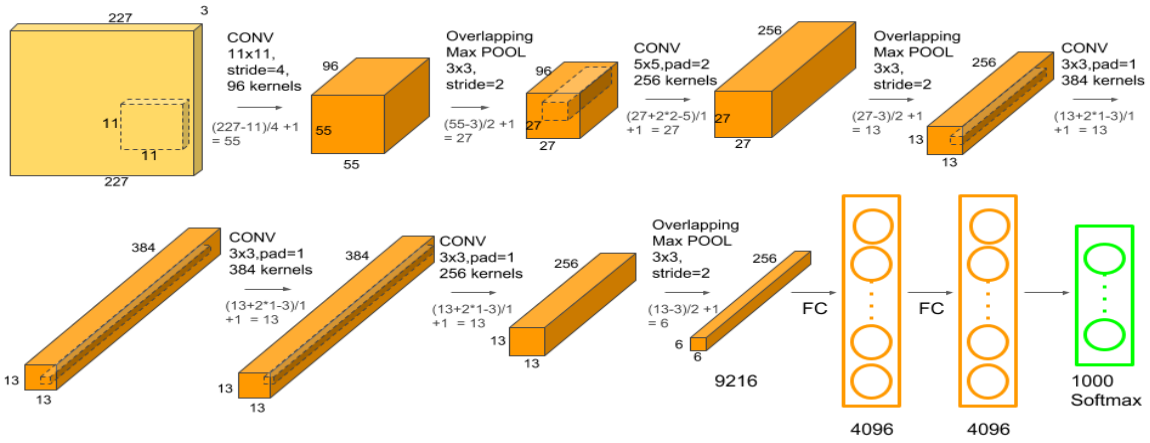


Figura 2.14. Arquitectura clásica de una CNN, se destacan los cambios de dimensiones por efecto de la stride de la convolución y la cantidad de kernels. En un Max Pool 3x3, se escoge el mayor valor de entre 9 píxeles en cada posición del filtro. El pad es la adición de píxeles en el margen de las feature maps o activaciones. Finalmente entra a una red neuronal "Fully Connected". Fuente: [73].

4.4. Generative Adversarial Networks (GANs)

Las **Redes Neuronales Generativas Antagónicas,** popularizadas por *Goodfellow et al. (2014)* [74], son arquitecturas basadas en un escenario teórico de juego en el que una RN llamada "**generador"** (*G*) debe competir contra un adversario, otra RN llamada "**discriminador"** (*D*).

El ***generador*** produce muestras $x = G(z; \vartheta^{(g)})$, donde ϑ son los parámetros y z una entrada de ruido gaussiano o una imagen a transformar junto con ruido. En la literatura también se generalizan las muestras generadas como p_{model} y la entrada como p_{data}. A su vez, el discriminador intenta distinguir entre las muestras extraídas del conjunto de entrenamiento (reales) y las muestras generadas por el *G* (falsas). En este proceso, el discriminador emite un valor de probabilidad dado por $D(x; \vartheta^{(d)})$, indicando la probabilidad de que *x* sea un ejemplo real de entrenamiento en lugar de una muestra falsa tomada del modelo generador.

Goodfelow et al. (2016) [43] refiere el aprendizaje de GANs como un juego de suma cero, en el que una función $v(\vartheta^{(g)}, \vartheta^{(d)})$ determina la recompensa del discriminador. El generador recibe $-v(\vartheta^{(g)}, \vartheta^{(d)})$ como penalización. Durante el aprendizaje, cada modelo intenta maximizar su propio beneficio.

La Ec. 15 expresa el juego de suma cero entre generador y discriminador, donde se intenta mejorar el conocimiento del discriminador mientras se quiere entrenar al generador para que el discriminador no haga lo propio. También se llama juego de *minmax* [p. 699, 43].

$$\theta^{(G)*} = arg\ min_{\theta^{(G)*}} max_{\theta^{(D)}} v(G, D)$$ **(Ec. 15)**

La Ec. 16 representa la expresión del intento por parte del discriminador de maximizar la probabilidad de distinguir una muestra de p_{data} cuando proviene de ahí (cosa que para que converja el entrenamiento tiene que empeorar). A su vez, el generador intentará maximizar el error del discriminador haciendo que aumente *d(x)* cuando la muestra es generada. Esto está expresado mediante el *cross-entropy error* [p. 699, 43].

$$v(\theta^{(g)}, \theta^{(d)}) = \mathbb{E}_{x \sim p_{data}} log\ d(x) + \mathbb{E}_{x \sim p_{model}} log\ (1 - d(x))$$ **(Ec. 16)**

La solución a un problema de optimización es un mínimo (local), un punto en el espacio de parámetros donde todos los puntos vecinos presentan un error mayor o igual. Este problema se describe como un equilibrio de Nash [75]. En este contexto, un equilibrio de Nash es una dupla ($\vartheta^{(D)}$, $\vartheta^{(G)}$), que es un mínimo local de $J^{(D)}$ con respecto a $\vartheta^{(D)}$ y un mínimo local de $J^{(G)}$ con respecto a $\vartheta^{(G)}$ (donde $J^{(D)}$ y $J^{(G)}$ son puntos del espacio multidimensional de la función de coste de la GAN, véase Ec. 17). Esto explica que no se puede permitir que el generador sea muy ineficaz y el discriminador muy bueno o viceversa [76].

4.4.1. Funciones de Coste en GANs

Cabe destacar que no existen unas restricciones concretas para las arquitecturas de GANs, ya que se pueden utilizar diversas funciones de perdida según sea conveniente, tanto para *D*, *G* o para la arquitectura entera de forma global.

4.4.1.1. *Cross-Entropy Loss* para discriminador (Adversarial)

La función de coste a minimizar del discriminador suele ser simplemente una *cross-entropy loss* binaria de la forma [p.21, 76]:

$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_{z} \log(1 - D(G(z)))$$ **(Ec. 17)**

4.4.1.2. *L1 y* $\mathbb{E}_z\, log(1 - D(G(z)))$ para generador

Basándonos en la anterior función de coste del discriminador, el generador G recibiría como función de pérdida "$-\ \mathbb{E}_z\, log(1 - D(G(z)))$". En la práctica se suele añadir una función de pérdida *L1* a la salida del generador si la *GAN* cuenta con *datasets* de datos emparejados, así se puede comparar el error píxel a pixel entre la imagen generada y la imagen objetivo.

4.4.1.3. *Perceptual* Loss

La *perceptual loss*, descrita en *Johnson et al. (2016)* [77], propone la utilización de la norma *L1* entre la imagen *target* que condiciona cada input del entrenamiento y la imagen generada mediante las activaciones extraídas de éstas al pasar por una CNN. Lo que se compara en la función son las activaciones tras las capas ocultas de una RN VGG16 (pre-entrenada en una base de datos tipo *Imagenet*) concatenada a la salida del generador. Como se muestra en la Figura 4.15, tanto la salida del *G* como el *target/s* entran en la VGG y se seleccionan activaciones (*feature maps*). Éstas son fruto de pasar filtros por ambas imágenes extrayendo así *features* o estructuras inherentes a las imágenes. Este método se utiliza para que la imagen generada pueda captar texturas y estilo, aunque no coincidan con la imagen target en posición exacta.

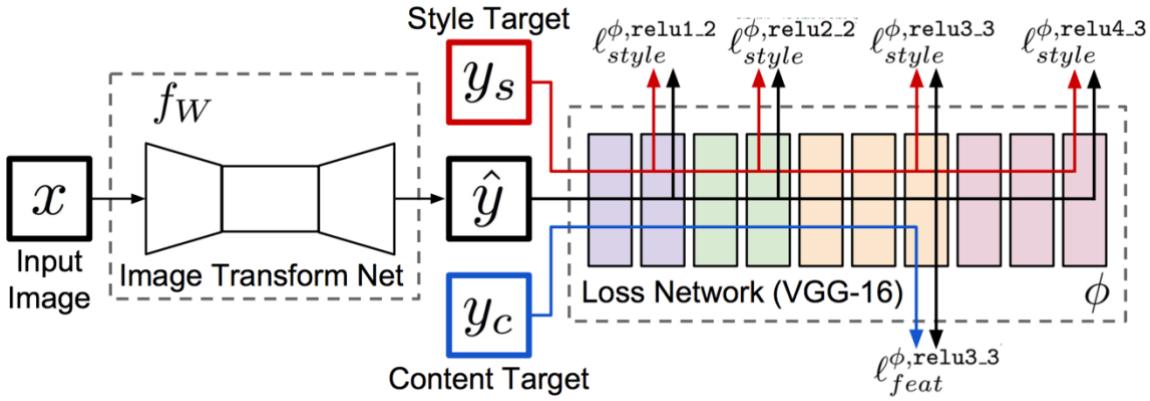


Figura 4.15. Diagrama de cómo se extraen los Feature Maps para calcular diferentes tipos de Perceptual Loss, como por ejemplo el Feature Reconstruction Loss, Style Reconstruction Loss, etc. Fuente: [77].

4.4.2. CycleGAN

La arquitectura *CycleGAN* [12] es de gran ayuda para cumplir el propósito de este trabajo. Esto es debido a que la entrada de dicha GAN no precisa de imágenes emparejadas (una para el conjunto *X* y otra para el conjunto *Y*). Es idóneo en nuestro caso el hecho de que, para aprender a extraer las características (*features)* necesarias para transformar del dominio *X* (imágenes de leucocitos tomadas desde un hospital *A*) al dominio *Y* (imágenes leucocíticas de un hospital *B* con características diferentes), no se necesite que para cada *X* exista una *Y* estrechamente correlacionada. Para que el conjunto {*X,Y*} fuera emparejado, las imágenes de las células tendrían que tener un equivalente en ambos microscopios pertenecientes a los diferentes hospitales con una posición lo más semejante posible. Esto es inviable para este trabajo, ya que se tendrían que localizar las mismas células y emparejarlas en dos artefactos ajenos el uno del otro y distribuidos en diferentes puntos del país, teniendo miles de GB por frotis.

4.4.2.1. Funciones de coste de *CycleGan*

En este apartado se introduce la idea de ciclo, donde se distinguen dos generadores *G* y *F.* Éstos se encargan de la transformación de un dominio en el otro de las imágenes en el entrenamiento. De forma homóloga, dos discriminadores (D_x y D_y) disciernen si la imagen pertenece a los conjuntos de datos o es generada por *G* o *F*. Cada ciclo refiere al proceso de transformar una muestra de un dominio en el otro y volver a reconstruirlo efectuando otra transformación de vuelta al dominio de origen.

Se introduce el ***cycle-consistency loss*** para modelizar el concepto anterior. Este concepto propone que si se pasa de un dominio a otro y viceversa se debería llegar a un punto cercano al de partida. Como se ve en la Figura 4.16 y se expresa en la Ec. 18:

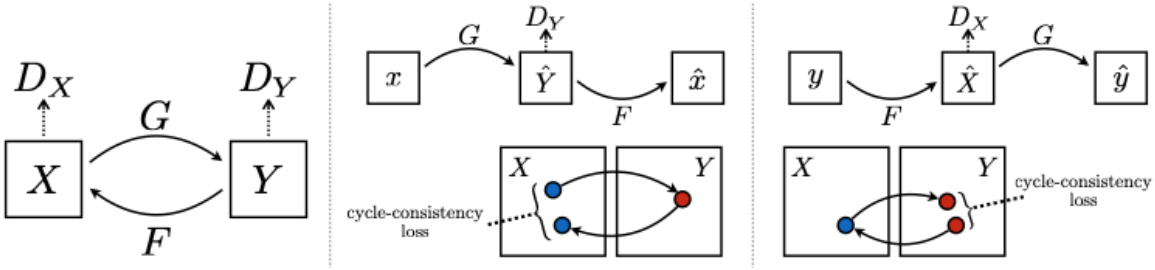


Figura 4.16. (Izquierda) Diagrama de la CycleGAN. (Medio) Forward cycle-consistency loss. (Derecha) Backward cycle-consistency loss. Fuente: [12].

- *Forward cycle-consistency loss*: $x \rightarrow G(x) \rightarrow F\,(G(x)) \approx x$
- *Backward cycle-consistency loss*: $y \rightarrow F\,(y) \rightarrow G(F\,(y))) \approx y$
- *Total cycle-consistency loss:*

$$L_{cyc}(G,F\,) = E_{x\sim p_{data\,(x)}}\left[\| F\,(G(x)) - x\,\|_1\right] + E_{y\sim p_{data\,(y)}}[\| G(F(y) - y\,\|_1] \textbf{ (Ec. 18)}$$

En la Ec. 18 se utiliza la norma *L1* (*MAE*) para comparar la imagen reconstruida, es decir, cuando vuelve a su dominio tras pasar por los dos generadores para muestras de X y de Y.

Acto seguido se expresan las ecuaciones de ***adversarial loss*** donde los generadores *G* (Ec. 19) y *F* (Ec. 20) intentan minimizar las funciones de pérdida mientras los discriminadores *Dy* (Ec. 19) y *Dx* (Ec. 20) las intentan maximizar.

$$L_{GAN}(G, D_y, X, Y\,) = E_{y\sim p_{data\,(y)}}\left[logD_y(y)\right] + E_{x\sim p_{data\,(x)}}[log\,(1 - D_y(G(x)))] \textbf{ (Ec. 19)}$$

$$Donde\ min_G\ max_{D_Y}\ L_{GAN}(G, D_Y, X, Y\,)$$

$$L_{GAN}(F, D_x, Y, X\,) = E_{x\sim p_{data\,(x)}}\left[logD_x(x)\right] + E_{y\sim p_{data\,(y)}}[log\,(1 - D_x(F(y)))] \textbf{ (Ec. 20)}$$

$$Donde\ min_F\ max_{D_X}\ L_{GAN}(G, D_X, Y, X)$$

La función objetivo de coste total queda definida como:

$$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X X, Y, X) + \lambda L_{cyc}(G, F) \textbf{ (Ec. 21)}$$

$$\mathrm{G}*, F* = argmin_{G,F}\, max_{D_X, D_Y} L(G, F, D_X, D_Y)$$

En este caso, la función de pérdida total de la *CycleGAN* se conforma como la fusión de las funciones de *adversarial loss* más la adición de la *cycle-consistency loss* regulada por un parámetro λ.

4.4.2.2. Generador y Clasificador en la *CycleGAN*

Los discriminadores que se utilizan son *PatchGANs* de 70 × 70, cuyo objetivo es clasificar si los parches de imagen superpuestos de 70 × 70 son reales o falsos. Esta arquitectura de discriminador a nivel de parche tiene menos parámetros que un discriminador de imagen completa y puede trabajar con imágenes de tamaño arbitrario de forma totalmente convolucional. Es propia de los discriminadores de la arquitectura *Pix2Pix*, donde se demuestra útil que la clasificación falso-real sea por zonas de la imagen y no la imagen completa.

En cuanto al generador, la arquitectura utilizada es de un *encoder-decoder*. Una RN de este tipo consta de dos etapas, una de abstracción por *downsample* o *encoding* (disminuye la dimensión de la entrada mediante convoluciones) y otra de *upsample* o *decoding* (vuelve a aumentar la dimensión por medio de deconvoluciones, la operación contraria a la convolución). En el *encoder* se llega a un cuello de botella mediante capas de convolución de una *ResNet* [78], que utiliza *skip-connections*. En el decoder tiene lugar la generación de la imagen transformada, desde el cuello de botella a la imagen *output*.

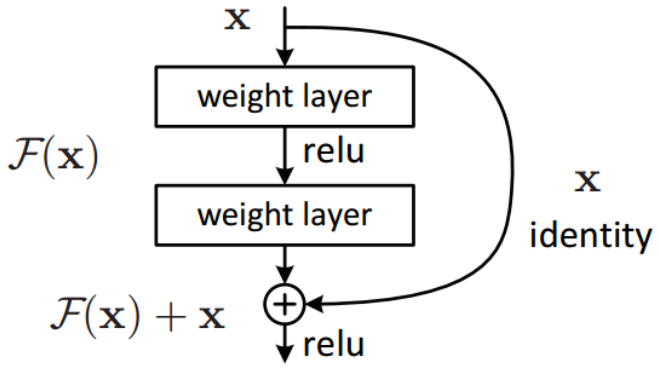


Figura 4.17. Resnet Block, tras la activación de una capa de convolución + ReLU se suma el input de la entrada, a veces se concatena. Esto es útil para que en una red generadora no se pierda la estructura de la imagen a transformar. Fuente: [78].

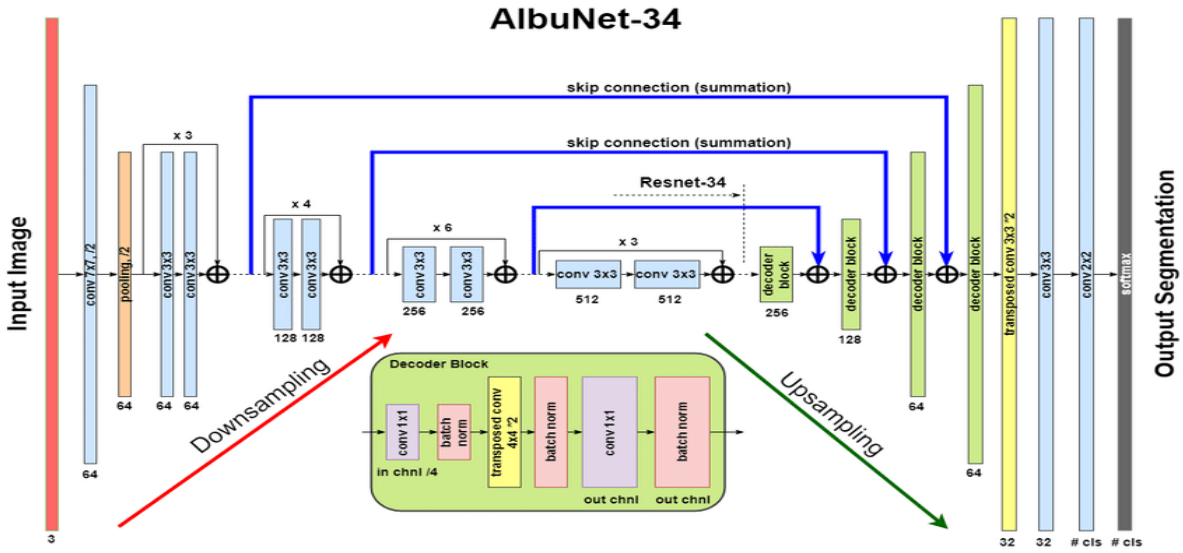


Figura 4.18. Ejemplo de una arquitectura ResNet encoder-decoder, las líneas en azul son Skip-Connections donde se suma el feature map de una capa a otro feature map. Nótense las etapas de Downsampling y Upsampling. Fuente: [79].

4.4.3. *ColorizationGAN*

La *ColorizationGAN* utilizada en este proyecto aprovecha funcionalidad de la *CycleGAN* para realizar entrenamientos con *datasets* no emparejados. No obstante, se basa en la arquitectura *Pix2Pix* de *image-to-image-translation.* Su proceso consiste en convertir los canales RGB de las imágenes a un canal *"L",* con el que realiza un mapeo a dos canales *Lab Color Space* (habiendo convertido las imágenes a escala de grises entremedio). A alto nivel, extrae *features* del conjunto de destino, pasa a escala de grises las imágenes a transformar y realiza el mapeo coloreando estas imágenes según las *features* extraídas [80], [p. 8 (fig. 9), 13].

5. Análisis del conjunto de datos

A continuación, se desglosará el conjunto de imágenes del Hospital Clínic y Can Ruti en su totalidad, llamados de aquí en adelante *dataset* o conjunto Clínic y *dataset* o conjunto Can Ruti.

5.1. *Dataset* del Hospital Clínic

En el caso del *dataset* Clínic, se contará con imágenes de 4 categorías (especificadas en la Tabla 1 y en la Figura 5.1). Son las de linfocito normal (*normal lymphocyte*), linfocito atípico (*atypical lymphocyte*), linfocito variante (*variant lymphocyte*) y blastos (*blast*). La categoría *atypical lymphocyte* resulta ser heterogénea en tanto que alberga distintas subcategorías dentro de sí (véase Tabla 2 y Figura 5.2).

Tabla 1. Desglose del Conjunto total de imágenes del Hospital Clínic.

	NORMAL LYMPHOCYTE	ATYPICAL LYMPHOCYTE	VARIANT LYMPHOCYTE	BLAST
Número de Imágenes	1086	3499	552	1240

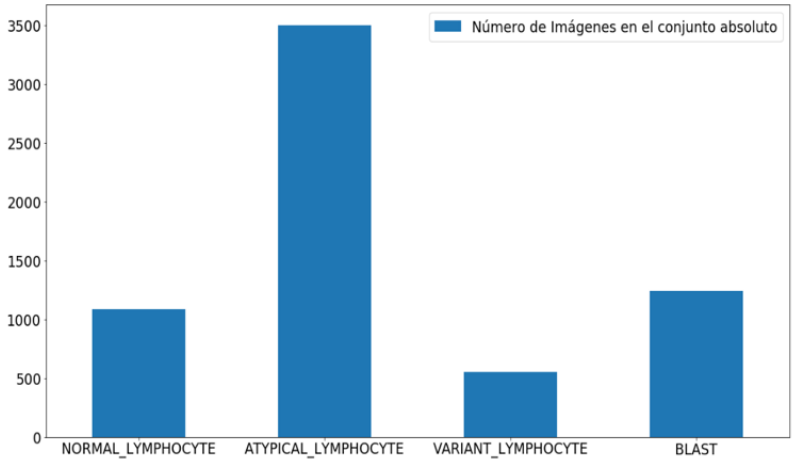


Figura 5.1. Diagrama de barras de las diferentes clases en el conjunto total Clínic. Fuente: Elaboración propia.

Tabla 2. Desglose de las subcategorías dentro de la clase atypical lymphocyte.

Subclase de ATYPICAL LIMPHOCYTE	N en la Categoría ATYPICAL LYMPHOCYTE
LZME (Linfoma de la zona marginal esplénico)	1583
LF (Linfoma folicular)	386
LCM (Linfoma de manto) INDOLENTE	590
LCM BLÁSTICO	121
LCM TÍPICO	477
LLC (Leucemia linfocítica crónica)	342

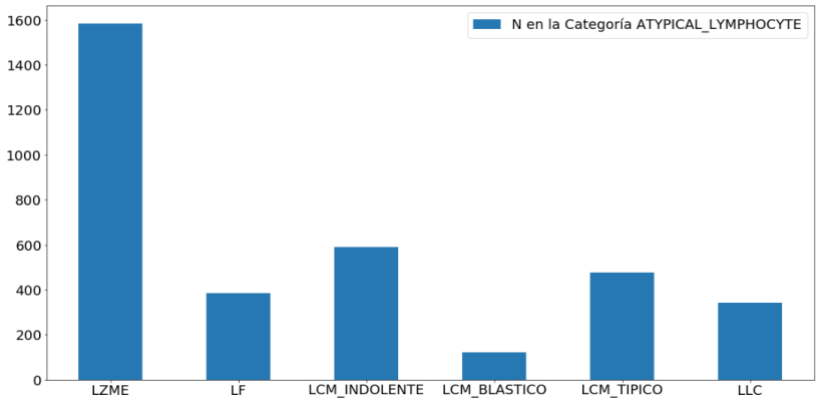


Figura 5.2. Diagrama de barras de las diferentes subclases de la clase atypical lymphocyte. Fuente: Elaboración propia.

Para la realización del *fine tuning* de los modelos CNN en el *dataset* Clínic se ha muestreado el conjunto Clínic (total), excluyendo la clase de linfocitos normales ya que en el conjunto Can Ruti no se dispone de ella. Véanse la Tabla 3 y la Figura 5.3.

Tabla 3. Desglose del conjunto Clínic para el fine tune de CNNs según categorías y según conjuntos de training, validation o testing.

Conjunto / Clase	Número de imágenes en el conjunto *Validation*	Número de imágenes en el conjunto *Testing*	Número de imágenes en el conjunto *Training*	Número de imágenes por clase en total
ATYPICAL LYMPHOCYTE	243	69	1026	1338
VARIANT LYMPHOCYTE	112	102	439	653
BLAST	227	93	866	1186
Total de imágenes por conjunto	582	264	2331	3177

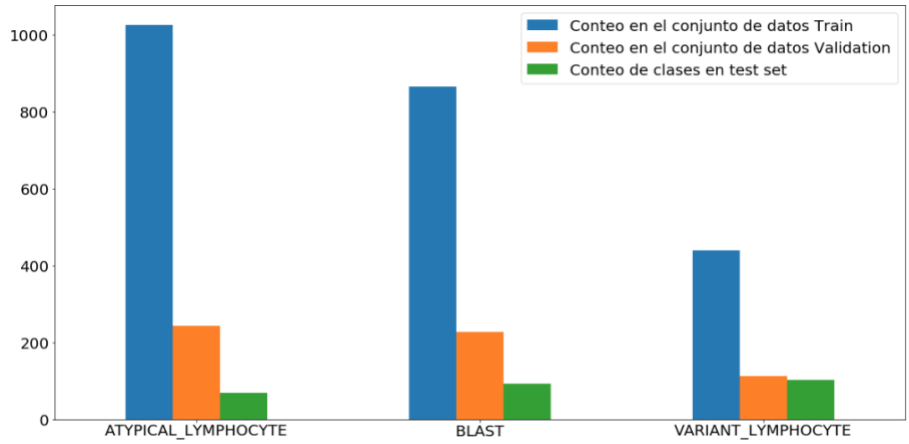


Figura 5.3. Diagrama de barras del conjunto Clínic para el fine tuning de CNNs según categorías y según conjuntos de training o testing. Fuente: Elaboración propia.

5.2. *Dataset* del Hospital Can Ruti

El *dataset* Clínic es bastante cuantioso en cuanto a imágenes. Por el contrario, el *dataset* Can Ruti cuenta con una distribución de datos menor en tamaño, especificada en la Tabla 4 y Figura 5.4. Debido a esto, se ha decidido no crear un *testing set* apartado y utilizar el conjunto de validación como *testing set* para calcular métricas de desempeño. Por lo tanto, en este *dataset*, los términos *validation set* y *testing set* serán intercambiables en los apartados de clasificación.

Este conjunto de imágenes participa en la clasificación por CNN (bloque 2 de este trabajo), además de formar parte del entrenamiento de las GANs para ser transformado acorde con los estándares del Hospital Clínic (bloque 1 de este trabajo). En el bloque 1, para entrenar las GANs transformadoras, se utilizará este dataset al completo, que posteriormente se transformará.

Debido a la transformación del bloque 1 posterior a los entrenamientos de *GANs* con los muestreos del Apartado 5.3, se han generado *datasets* Can Ruti *Fake.* Estos se han clasificado también con *CNNs* en el bloque 2. Estos *datasets* Can Ruti *Fake* se explicarán en el apartado de metodología (6.1).

Lo ideal hubiera sido contar con un conjunto de *testing* totalmente ajeno al entrenamiento (para el *fine tuning* del Apartado 6.2.2.2), para así no crear ningún tipo de sesgo. Esto es porque si se evalúan las métricas usando sólo un conjunto de validación, el hecho de parar el entrenamiento cuando se considera que la métrica de exactitud del *validation set* es suficientemente buena, crea un pequeño sesgo. Teniendo un conjunto de *testing* se puede evaluar el modelo con datos totalmente desconocidos para él.

Tabla 4. Desglose del conjunto Can Ruti según clase y si pertenecen al conjunto de training o validación.

Conjunto / Clase	Número de imágenes en el conjunto *Training*	Número de imágenes en el conjunto *Validation*	Total de imágenes por clase
ATYPICAL LYMPHOCYTE	22	25	47
VARIANT LYMPHOCYTE	27	30	57
BLAST	43	37	80
Total de imágenes por conjunto	92	92	184

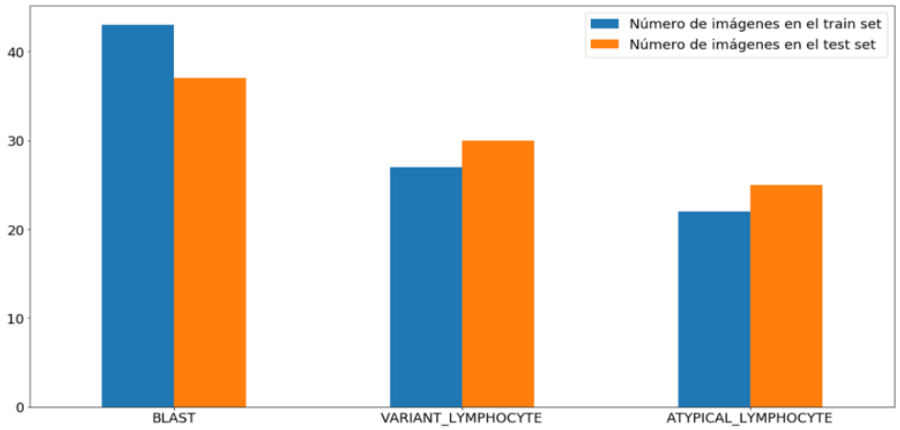


Figura 5.4. Diagrama de barras del conjunto Clínic para el fine tuning de CNNs según categorías y según conjuntos de training o testing. Fuente: Elaboración propia.

5.3. Muestreo de imágenes para el entrenamiento de GANs (transformación del *dataset* Can Ruti)

5.3.1. Muestreo Can Ruti-Clínic no separado por clases (para CycleGAN y *ColorizationGAN*)

Con el fin de entrenar las *GANs* para realizar transformaciones en el conjunto Can Ruti, se necesita un input en paralelo de las muestras de éste y del *dataset* Clínic enfrentadas entre sí. En los experimentos se ha comparado la generación de resultados en *CycleGAN* y *ColorizationGAN*.

Se ha optado por hacer dos muestreos aleatorios diferentes de las imágenes del Clínic para el entrenamiento de cada red *GAN*. Estos muestreos del Clínic se han enfrentado al *dataset* Can Ruti al completo. Dichas distribuciones de datos se especifican en la Tabla 5 (número de imágenes por tipo de *set* vs arquitectura GAN) y en la Figura 5.5. El conjunto de *testing* contiene exactamente las mismas

imágenes que el conjunto de *training* en el caso del Hospital Can Ruti, pero en el conjunto Clínic los dos *testing sets* contienen imágenes diferentes.

Los *training sets* han servido para entrenar la GAN y los *testing sets* para realizar las transformaciones (aunque como se ha dicho el *testing* set y el *training set* de can Ruti son idénticos).

Tabla 5. Desglose de los conjuntos de imágenes utilizados para el entrenamiento de las GANs transformadoras según dataset, arquitectura GAN y hospital.

Arquitectura GAN / Conjunto de Datos	ColorizationGAN (Normal Lymphocyte en Clínic)	CycleGAN (Atypical + Blast+ Variant en Clínic)
Conjunto *Training* Clínic	1086	3000
Conjunto *Training* Can Ruti	184	184
Conjunto *Testing* Clínic	184	184
Conjunto *Testing* Can Ruti	184	184

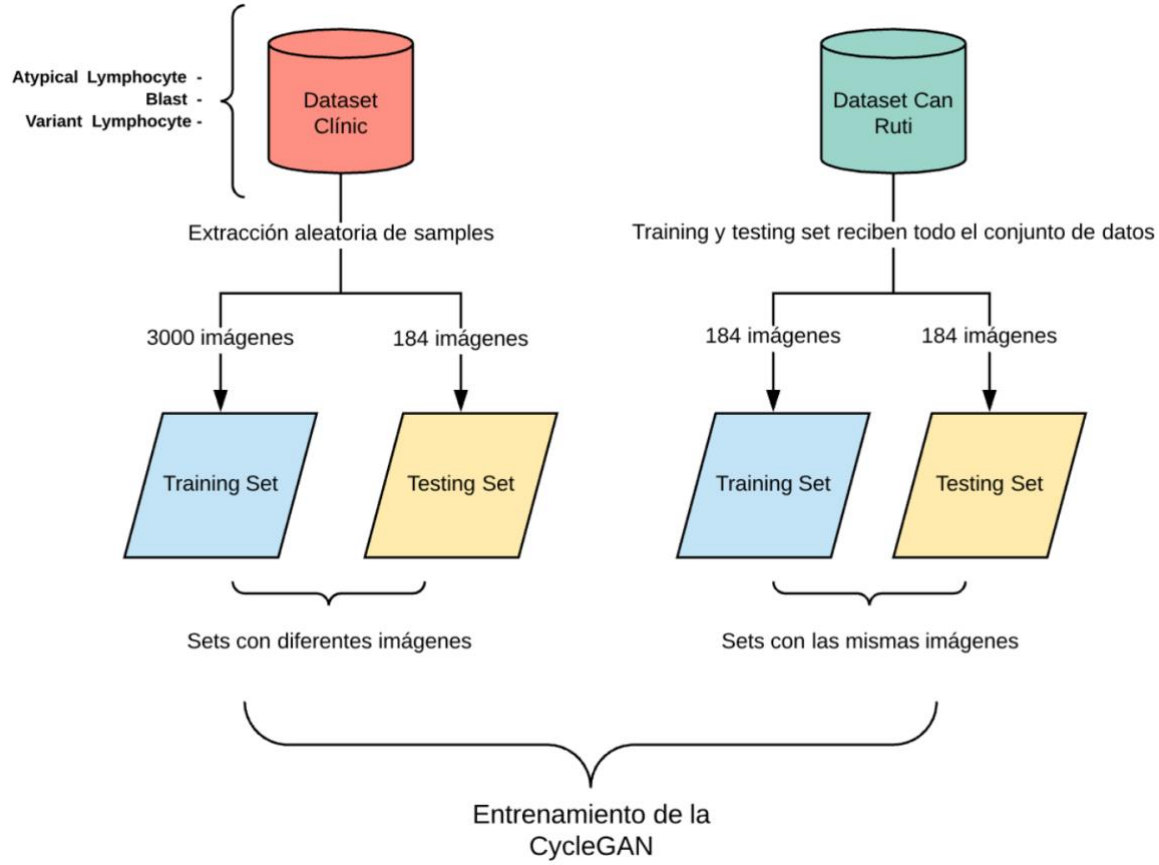


Figura 5.5 Diagrama de selección del muestreo no separado por clases para el entrenamiento de la CycleGAN. Fuente: Elaboración propia.

5.3.2. Muestreo Can Ruti-Clínic separado por clases (para CycleGAN)

El muestreo explicado en este punto es diferente al explicado en el punto anterior. La finalidad de éste es entrenar tres modelos diferentes de la arquitectura *CycleGAN* de manera que cada uno se centre en una clase por separado. Para ello, en cada entrenamiento se han utilizado muestras de solo una clase, pero de los dos hospitales enfrentados (ver Figura 5.6).

- Input entrenamiento 1: *Atypical Lymphocyte* (*Can Ruti)* vs *Atypical Lymphocyte* (*Clínic*)
- Input entrenamiento 2: *Variant Lymphocyte* (*Can Ruti*) vs *Variant Lymphocyte* (*Clínic*)
- Input entrenamiento 3: *Blast* (*Can Ruti*) vs *Blast* (*Clínic*)

Este procedimiento no podría tener lugar en un conjunto de *testing* real ya que no se conocerían las clases a priori. Aun así, se ha creado este muestreo separado por clases para evaluar un desempeño de clasificación entrenando la red transformadora de una forma ideal. Después de realizar la transformación se juntarán todas las imágenes generadas (*fake)* en un nuevo *dataset.* Los conjuntos de *testing* contienen las mismas imágenes que los conjuntos de *training* en el caso del hospital Can Ruti, pero en el conjunto Clínic lo *trainings sets* y *testing sets* contienen imágenes diferentes. De aquí en adelante esta distribución de datos recibirá el nombre de *Dataset Can Ruti Separate.*

Tabla 6. Desglose de los conjuntos de imágenes utilizados para el entrenamiento de las GANs transformadoras según dataset y clase.

Conjunto de Datos / Clase	Conjunto Training Clínic	Conjunto Training Can Ruti	Conjunto Testing Clínic	Conjunto Testing Can Ruti
ATYPICAL LYMPHOCYTE	2800	47	47	47
VARIANT LYMPHOCYTE	552	57	57	57
BLAST	900	80	80	80

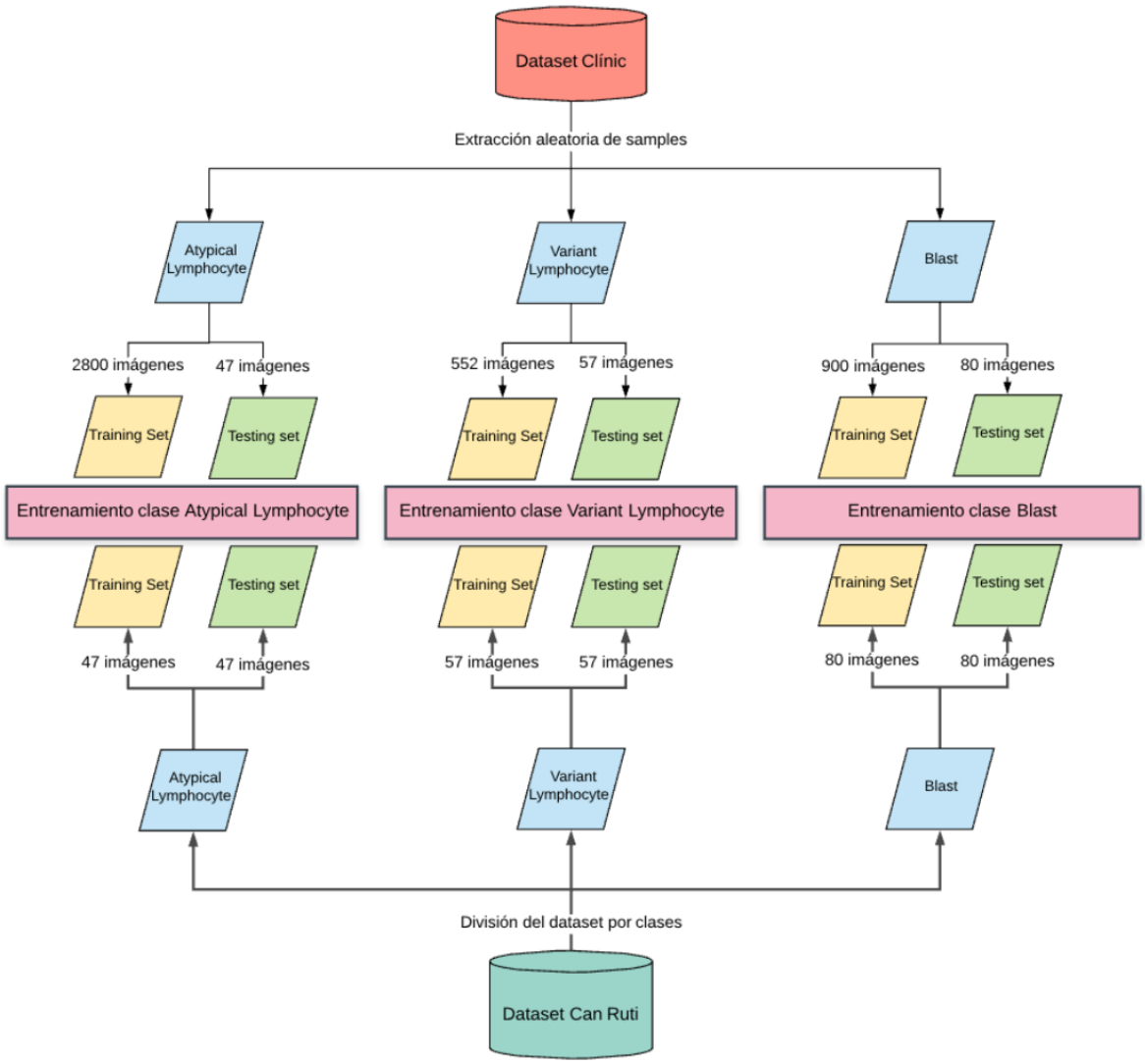


Figura 5.6. Diagrama de la selección del muestreo separado por clases para el entrenamiento de la CycleGAN. Se llamará Dataset Separate. Fuente: Elaboración propia.

5.4. Muestreo de imágenes para el entrenamiento de *CycleGAN* para la transformación de las imágenes del conjunto Clínic entre clases (Bloque 3 de la metodología)

Se han seleccionado tres muestreos de imágenes del conjunto Clínic para entrenar la *CycleGAN*. Se han realizado tres ensayos con cada muestreo, en todos ellos habrá un subconjunto de linfocitos normales y el otro será de una categoría diferente (LZME, LCM y blastos). Los tamaños de cada muestreo se

presentan en la Tabla 7 y en la Figura 5.7. Los experimentos serán los enumerados a continuación, pertenecientes al bloque 3 de este trabajo:

- Linfocitos normales vs LZME
- Linfocitos normales vs LCM
- Linfocitos normales vs Blastos

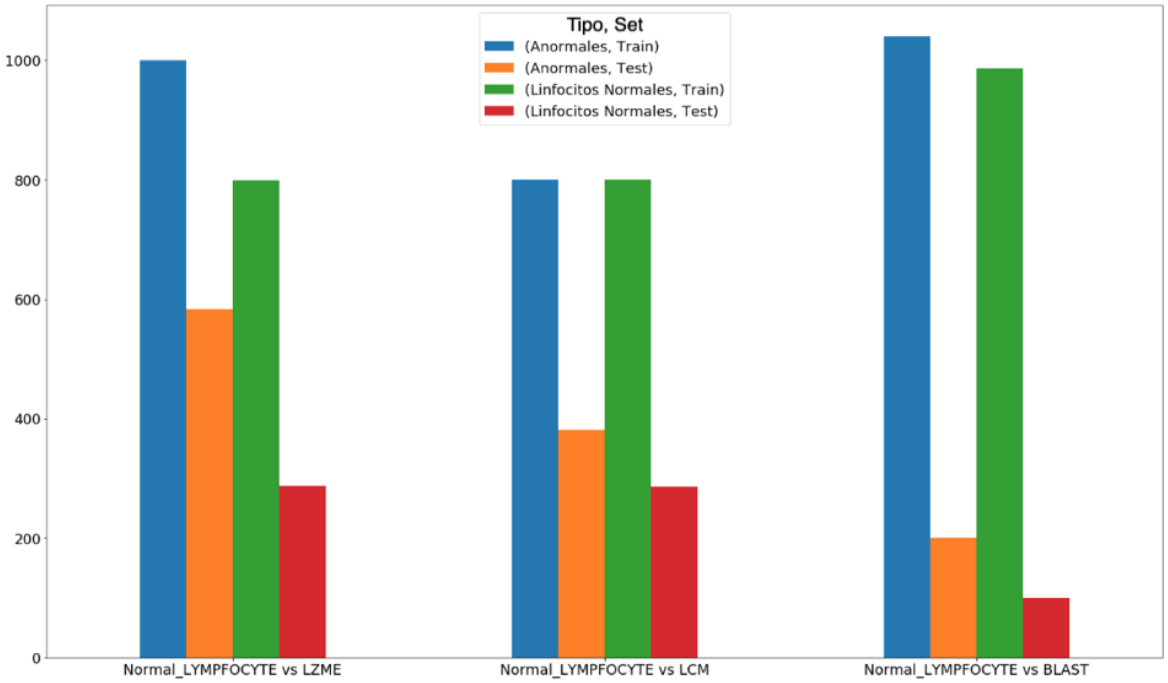


Figura 5.7. Diagrama de barras de los muestreos de imágenes según experimento con CycleGAN y subconjunto de la distribución (training y testing). Fuente: Elaboración propia.

Tabla 7. Desglose de los muestreos de imágenes según experimento con CycleGAN y subconjunto de la distribución (training y testing).

Hospital / Ensayo	Anormales		Linfocitos Normales	
	Training	Testing	Training	Testing
Normal LYMPHOCYTE vs LZME	1000	583	799	287
Normal LYMPHOCYTE vs LCM	800	381	800	286
Normal LYMPHOCYTE vs BLAST	1040	200	987	100

6. Metodología empleada y resultados

La metodología de investigación de este TFE consta de tres bloques que coinciden con los tres objetivos principales explicados anteriormente en el Apartado de 2.1. En los siguientes subapartados de este capítulo se explicará la metodología seguida en cada bloque y se presentarán los consiguientes resultados individualmente, ya que cada bloque da lugar a resultados por sí mismo.

6.1. Bloque 1: transformación de imágenes de Can Ruti mediante GANs

Como se muestra en el diagrama de la Figura 6.1, el proceso de conversión de las imágenes del conjunto Can Ruti a los estándares del Hospital Clínic consta de un *pipeline* de 4 pasos bien diferenciados. Se enumerarán a continuación y se explicarán más detalladamente en los siguientes subapartados:

1. Creación de los *dataroots* donde disponer los conjuntos de *training* y *testing set* para cada experimento con las *GANs*.
2. Selección del *preset* de parámetros de entrenamiento de las *GANs* y realización de los entrenamientos (experimentos) con los modelos *ColorizationGAN* y *CycleGAN.*
3. Testeo o generación de resultados de imágenes Can Ruti *Fake* con las dos arquitecturas. Recuérdese que el conjunto de *training* y de *testing* en el conjunto Can Ruti son idénticos.
4. Valoración cualitativa de las imágenes generadas y comparación de éstas (Can Ruti *Fake*) con el conjunto Clínic y entre las diferentes variantes de *datasets fake.*

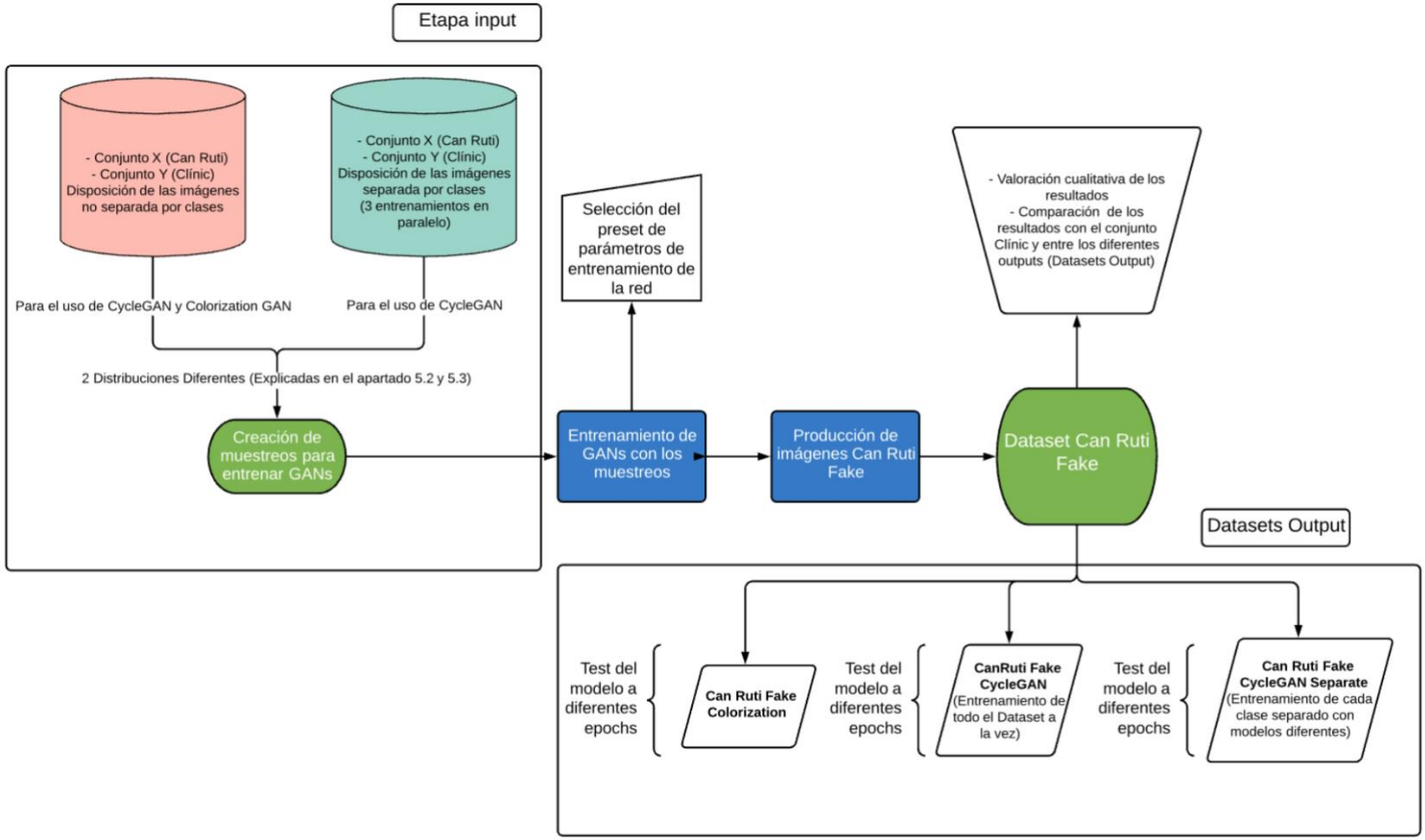


Figura 6.1. Diagrama del Bloque 1: transformación de imágenes de Can Ruti con GANs. Fuente: Elaboración Propia.

6.1.1. Creación de los *dataroots*

Este punto describe la creación del **muestreo** en los dos hospitales para crear los dos conjuntos especificados en los Apartados 5.3.1. y 5.3.2. Como se ha explicado habrá dos diferentes muestreos, uno separado por clases y otro con todas las clases juntas. A continuación, se muestra la disposición de ambos muestreos organizados en carpetas dentro del *docker* (Figura 6.2, Figura 6.3 y Figura 6.4):

Figura 6.2. Dentro de la carpeta CanRuti_Clinic_Nonormal se encuentran los subconjuntos ya especificados de training y testing de Can Ruti y Clinic no separados por clase. "A" simboliza "Can Ruti" y "B" simboliza Clínic. Fuente: Docker propio.

Figura 6.3. Dentro de la carpeta Dataset_Separate se disponen tres carpetas (una para cada clase) dentro de las cuales hay una disposición igual a la Figura 6.2. Fuente: Docker propio.

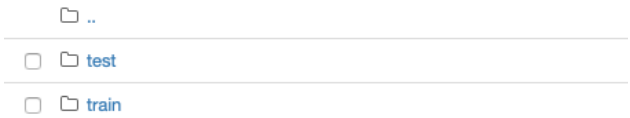


Figura 6.4. Disposición de las carpetas para la ColorizationGAN. Para el entrenamiento de esta red se han de explicitar las carpetas de esta forma: `train` (conjunto A, Can Ruti) y `test` (conjunto B, Clínic). Fuente: Docker propio.

La disposición de las imágenes entregadas por parte del Hospital Clínic está estructurada tal como se muestra en la Figura 6.5, para extraer este *folder tree* se ha utilizado la función **`list_files`**, visible en el Anexo (A.1, Fragmento de código 16):

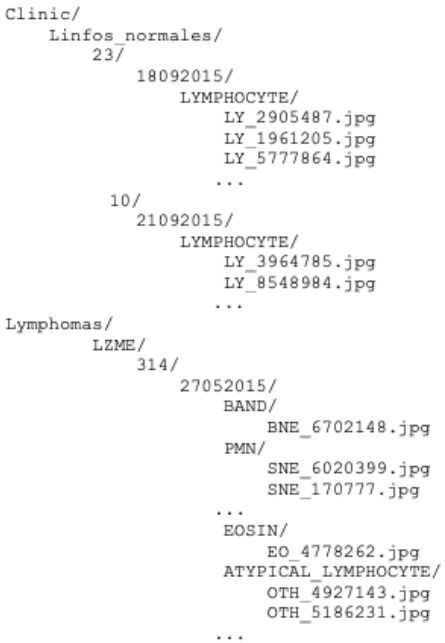


Figura 6.5. Folder tree realizado mediante la función "list_files" en la carpeta `Dataset_Clinic` (base de datos cruda). Los datos están organizados en orden paciente -> frotis -> tipo de célula. Los directorios de interés de los cuales se obtendrán las imágenes serán "ATYPICAL_LYMPHOCYTE", "BLAST", "VARIANT_LYMPHOCYTE" y "LYMPHOCYTE". Fuente: Docker propio.

Para seleccionar las imágenes necesarias dentro de cada subdirectorio de esta base de datos, se ha creado la función **`walk_dirs_recursively`**, que recorrerá todo el *dataset* dentro de la carpeta **`Dataset_Clinic`** y añadirá las imágenes a un diccionario (**`dic`**). Acto seguido se recorrerá este diccionario **`dic`** para copiar las imágenes y pegarlas a las carpetas que se utilizarán como *dataroot* de *training* y *testing* de las *GAN* (ver Fragmento de código 1). Esta copia de archivos se realizará mediante la función del módulo de *Python* **Os** llamada **`os.copyfile`**. Para reunir las imágenes del Hospital Can Ruti se utilizará un proceso análogo con las mismas funciones, customizándolas para copiarlas a carpetas diferentes.

```
1.  dic = {'NORMAL_LYMPHOCYTE':[],
2.         'ATYPICAL_LYMPHOCYTE':[],
3.         'VARIANT_LYMPHOCYTE':[],
4.         'BLAST':[]
5.        }
6.  Dataset_Clinic = 'Carpeta donde se encuentra la base de datos cruda facilitada
    por Hospital Clinic'
7.  # Función para recorrer el dataset Clínic y añadir las rutas de las imágenes al
     diccionario dic
8.
9.  def walk_dirs_recursively(root_dir):
10.     root_dir = os.path.abspath(root_dir) # Coge el primer directorio
11.
12.     for item in os.listdir(root_dir):
13. # lista los diferentes directorios de este
14.         item_full_path = os.path.join(root_dir, item)
15. # crea el formato apropiado para las nuevas rutas
```

```
        if os.path.isdir(item_full_path):
# si es una carpeta vuelve a utilizarla función de forma recursiva
            walk_dirs_recursively(item_full_path)
        else:  # si encuentra las palabras clave las añade al diccionario
            if '/Linfos_normales/' in item_full_path:
                dic['NORMAL_LYMPHOCYTE'].append(item_full_path)
            if '/ATYPICAL_LYMPHOCYTE/' in item_full_path:
                dic['ATYPICAL_LYMPHOCYTE'].append(item_full_path)
            if '/VARIANT_LYMPHOCYTE/' in item_full_path:
                dic['VARIANT_LYMPHOCYTE'].append(item_full_path)
            if '/BLAST/' in item_full_path:
                dic['BLAST'].append(item_full_path)

walk_dirs_recursively(Dataset_Clinic)
```

Fragmento de código 1. Algoritmo de obtención de imágenes en la base de datos de imágenes Clínic (Cruda).

6.1.2. Entrenamiento de las GAN

6.1.2.1. Entrenamiento de la CycleGAN

Para entrenar la *CycleGAN* con los dos *datasets* se ha optado por los valores para entrenamientos de dimensiones similares recomendados por Jun-Yan Zhu, desarrollador principal de la arquitectura. Éstos se pueden ver en su repositorio de *Github* [80]. Los parámetros a destacar son:

- Utilización de *mini-batches* de 1 sola imagen
- Opción guardar los pesos del modelo cada 5 *epoch*
- *Dataset* no alineado (no hay pares equivalentes entre los conjuntos de los dos Hospitales)
- **`load_size`** (escalamientos de las imágenes) y **`crop`** (recorte) por defecto (286 y 256 pixeles respectivamente)
- 200 *epochs* de entrenamiento (**`n_epochs = niter + niter_decay`**)
- **`“ngf”, “ndf” = 64`** (Número de filtros en la última capa del discriminador y la primera del generador)

Para ver un ejemplo de *preset* de parámetros para el entrenamiento de la *CycleGAN* en los dos *datasets* de entrenamiento de Can Ruti (*Dataset Separate y* el no separado por clases) visite el Anexo (A.2).

- **Caso del muestreo no separado por clases:** Como se explicará posteriormente en el apartado de resultados de este bloque, transformar las imágenes con 200 *epochs* de entrenamiento causa modificaciones morfológicas no deseadas en las células. Por lo tanto, se han probado distintos ensayos de generación de imágenes Can Ruti *Fake* a *epochs* de entrenamiento más

tempranos. Ha sido el modelo a 15 *epochs* el que causaba resultados más favorables en clasificaciones. En total se ha testeado la transformación del modelo a 10, 15, 20, 25 y 200 epochs.

- **Caso del muestreo separado por clases (*epochs* óptimos):** La ejecución de la producción de imágenes Can Ruti *Fake* se ha realizado con los 3 modelos (uno para cada clase) entrenados a 200 *epochs* y también con cada modelo entrenado a sus ***epochs* "óptimos"**. Esto es, para cada clase se ha tanteado la transformación de imágenes con *epochs* más tempranos hasta que se han encontrado resultados que solo cambien color y no morfología celular. Los *epochs* escogidos para cada clase serán los siguientes:
    - Variant Lymphocyte: 10 *epochs*
    - Blast: 15 *epochs*
    - Atypical Lymphocyte: 20 *epochs*

La instrucción para entrenar la *CycleGAN* a introducir en el terminal será la siguiente:

```
python train.py --dataroot /shared/{directorio /Dataset a escoger} --name {Nombre a
escoger} --model cycle_gan
```

Fragmento de código 2. Instrucciones para activar el script de entrenamiento de la CycleGAN. Fuente: [80].

```
python test.py --dataroot {directorio/Dataset a escoger} --name {Nombre a escoger} --
model cycle_gan --results_dir {directorio/depositar/resultados} --num_test {# imágenes
de test} --epoch {# Epochs a cargar en el modelo}
```

Fragmento de código 3. Instrucciones para activar el script de test de la CycleGAN. Fuente: [80].

6.1.2.2. Entrenamiento de la *ColorizationGAN*

Los criterios de entrenamiento de la *ColorizationGAN* coinciden con los de la *CycleGAN*, pero hay aspectos que cambian. Hay que destacar que esta *GAN* realiza una conversión a escala de grises de las imágenes previamente a realizar la transformación de color (en la que se centra esta arquitectura, dejando de lado transformaciones morfológicas). Por eso se insta el parámetro **`dataset_mode=colorization`**. Al igual que en los entrenamientos anteriores, se ha probado el testeo de transformación a diferentes *epochs* de entrenamiento (20, 50, 70 y 200 *epochs*).

Al ver que los resultados no varían de forma notable se utilizará el resultado de los 200 para el bloque 2 de clasificación. Para ver un ejemplo *Preset* de parámetros para el entrenamiento de la *ColorizationGAN* con sus dos *datasets* de entrenamiento visite el Anexo (A.2).

```
python train.py --dataroot {directorio/Dataset a escoger} --name {Nombre a escoger} --model colorization --dataset_mode colorization --input_nc 1 --output_nc 2
```

Fragmento de código 4. Instrucciones para activar el script de entrenamiento de la ColorizationGAN. Fuente: [80].

```
python test.py --dataroot {directorio/Dataset a escoger} --name {Nombre a escoger} --model colorization --results_dir {directorio/depositar/resultados} --dataset_mode colorization --input_nc 1 --output_nc 2 --num_test {# imágenes de test} --epoch {# Epochs a cargar en el modelo}
```

Fragmento de código 5. Instrucciones para activar el script de test de la ColorizationGAN. Fuente: [80].

6.1.3. Resultados del Bloque 1

Los resultados que se mostrarán a continuación se pueden visualizar en formato **.png** en el repositorio de *Github* de este TFE [2] (directorio **/Notebooks_Results_TFG/results_TFG/Bloque_1**). Dentro de las carpetas de cada experimento hay distintos directorios donde se almacenan las imágenes (carpetas **images**) según el *epoch* de entrenamiento del modelo al transformar. Las imágenes con extensión **"_real_B"** y **"_fake_A"** son las Can Ruti originales y las Can Ruti transformadas respectivamente. Por otro lado, las imágenes **"_real_A"** y **"_fake_B"** son las imágenes Clínic originales y Clínic transformadas respectivamente.

Se puede decir a simple vista que los resultados que han emulado mejor los estándares Clínic son los obtenidos a partir de utilizar la *CycleGAN* entrenada en 10, 15 y 20 *epochs* (ver Figura 6.6). Nótese que el test a 10 *epochs* muestra un patrón de cuadrícula de puntos (no se ha conseguido suficiente nitidez). Se muestra en la cuadrícula comparativa de la Figura 6.7 un ejemplo de cada clase del conjunto Can Ruti transformado mediante las GANs en cada experimento. Hay más imágenes de resultados que se pueden visibilizar en el Anexo (B.2).

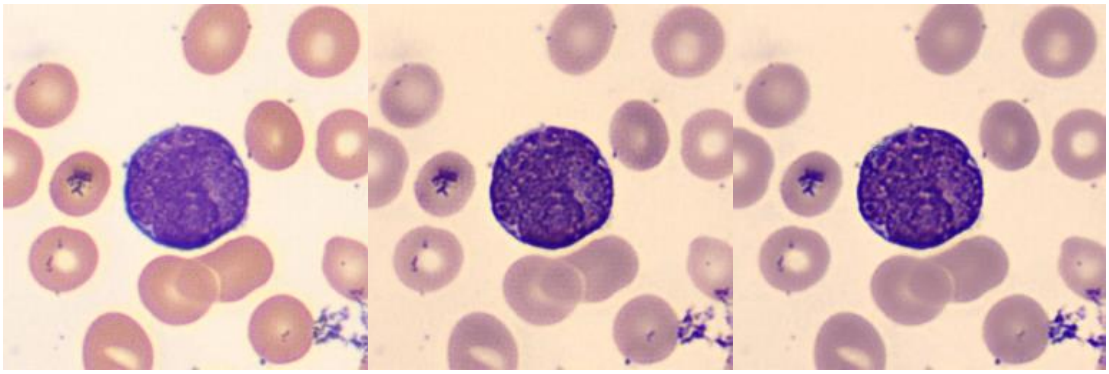


Figura 6.6. Comparación del test de la CycleGAN entrenada a 10 Epochs (Centro) y a 15 Epochs (Derecha). Nótese el patrón de cuadrícula de puntos en la imagen central. La imagen izquierda es la imagen original tomada en Can Ruti. Fuente: Dataset Can Ruti e imágenes generadas.

Como resultados no deseados se destacan las modificaciones morfológicas de los experimentos con *CycleGAN* a 25 y 200 *epochs* con el entrenamiento no separado por clases (5ª y 6ª columna de la Figura 6.7) y el de 200 epochs con el *Dataset Separate* (*8ª* columna de la Figura 6.7). En estas imágenes se produce un pervertimiento en la imagen ya que la *GAN* añade una información irreal. En el caso del experimento con la *ColorizationGAN*, hay que destacar que no se ha conseguido un resultado favorable en cuanto a color, ya que el tono sigue siendo de un amarillo similar a las imágenes originales e incluso el citoplasma ha tomado un color más rojizo.

Por lo que respecta a la transformación del *Dataset Separate* (Can Ruti) con *epochs* óptimos (7ª columna de la Figura 6.7), hay que destacar que hay pocos cambios morfológicos, pero los hay, aunque

son sutiles. Como se ha dicho, los *epochs* de entrenamiento óptimos son los que han logrado unos resultados más favorables en un tanteo de transformación a diversos *epochs* (explicado en el Apartado 6.1.2.1). Estos han podido cambiar ligeramente una célula según los patrones de su misma clase del conjunto Clínic (cosa que para una red clasificadora entrenada en Clínic sería más fácil de reconocer, aunque no conserve la forma original).

Esto ha ocurrido de forma más evidente en el caso de la prueba de este mismo *dataset* a 200 *epochs* (9ª columna de la Figura 6.7). En el caso de los linfocitos variantes, al no disponer de una muestra más grande en el conjunto Clínic y Can Ruti, los resultados de la conversión del Dataset Can Ruti *Separate* (*epochs* óptimos) contienen bastante ruido (2ª fila 8ª columna de la Figura 6.7).

A continuación, se enumeran las variantes de *datasets* del conjunto Can Ruti existentes al final de este bloque, contando tanto la original como las variantes *fake.*

- Can Ruti Original
- Can Ruti *Fake CycleGAN 15 epochs (No Dataset Separate)*
- Can Ruti *Fake CycleGAN 20 epochs (No Dataset Separate)*
- Can Ruti *Fake CycleGAN 200 epochs (No Dataset Separate)*
- Can Ruti *Fake CycleGAN (Dataset Separate) epochs óptimos*
- Can Ruti *Fake CycleGAN (Dataset Separate) 200 epochs*
- Can Ruti *Fake ColorizationGAN 200 epochs*

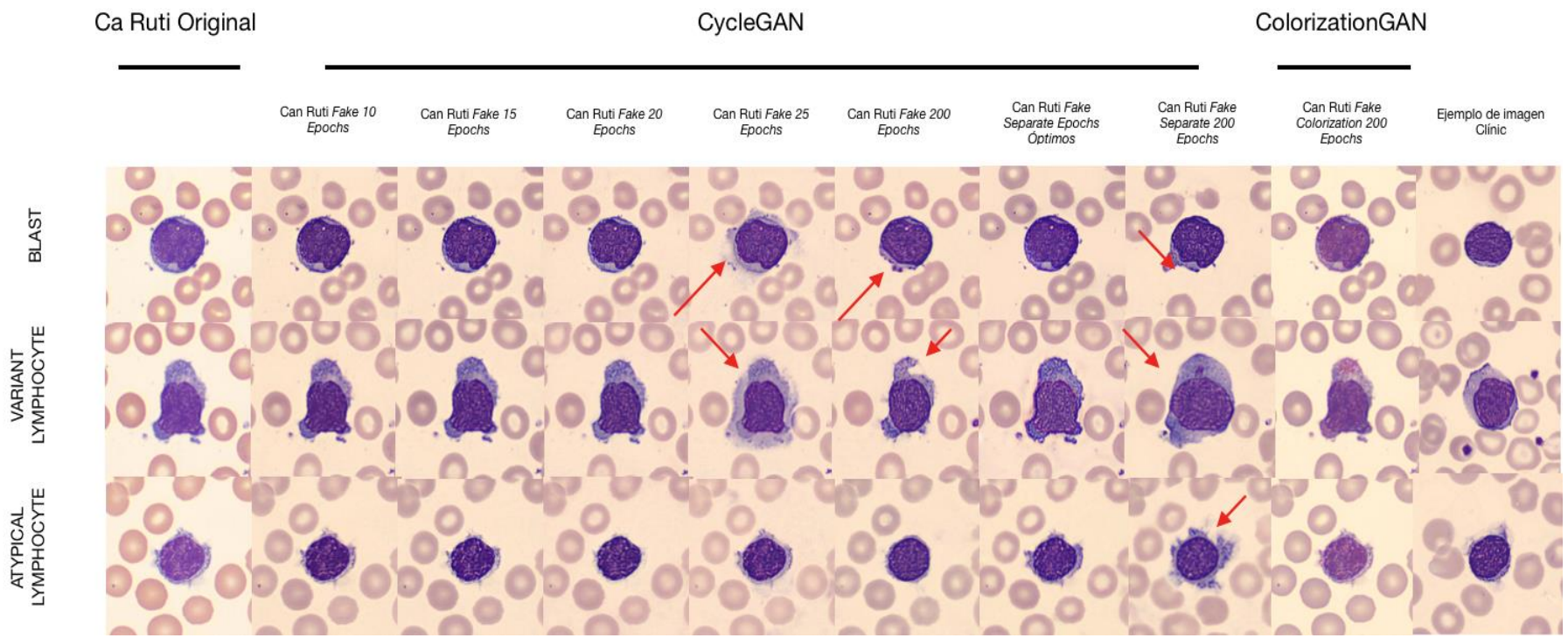


Figura 6.7. Cuadrícula comparativa del resultado de las transformaciones. Eje vertical: tipo de célula; eje horizontal: experimento realizado. Las flechas rojas muestran algunos de los cambios no deseados. Fuente: Elaboración Propia.

6.2. Bloque 2: Clasificación automática de imágenes de células de sangre periférica (originales y *fake*) del Hospital Can Ruti mediante CNNs

En este bloque de experimentos se realizarán las clasificaciones mediante CNNs de las imágenes del conjunto Can Ruti original y de los *datasets fake* generados en el bloque 1. Los pasos que conforman el *pipeline* de este bloque se resumen a continuación (ver diagrama de la Figura 6.8):

1. Obtención de las arquitecturas preentrenadas en Imagenet y del modelo *ResNet34_Resample_NoSIND* (modelo entrenado en un *dataset* del hospital Clínic mayor al disponible en este TFE). Los modelos serán los siguientes:
    - *ResNet 18*
    - *ResNet 18 Focal Loss*
    - *ResNet 34*
    - *Resnet 34 Focal Loss*
    - *ResNet_34_Resample_NoSIND*
2. Elaboración de una función de pérdida llamada *Focal Loss* para mejorar el entrenamiento con *datasets* desbalanceados en clases
3. *Fine tuning* usando el conjunto Clínic en los modelos exceptuando *ResNet34_Resample_NoSIND,* ya que ha tenido un *fine tuning* en un *dataset* mayor.
4. Test de clasificación en los *datasets* Can Ruti de las CNNs, tanto en el conjunto original como en los conjuntos de imágenes generados en el bloque 1 (registro de las métricas de *accuracy*, precisión y matrices de confusión).
5. *Fine tuning* de estos modelos usando los respectivos *datasets* (originales y generados).
6. Test de clasificación en los *datasets* respectivos (originales y generados) después del fine tuning realizado en el punto (5). Por ejemplo, la red ResNet 34 clasifica el conjunto de validación del *dataset* Can Ruti *Fake* 20 epochs, y luego realiza un fine tuning en el conjunto de *training* de este *dataset* y vuelve a clasificar el conjunto de validación.
7. Tabular y discutir los resultados de los modelos antes y después del fine tuning en el hospital Can Ruti.

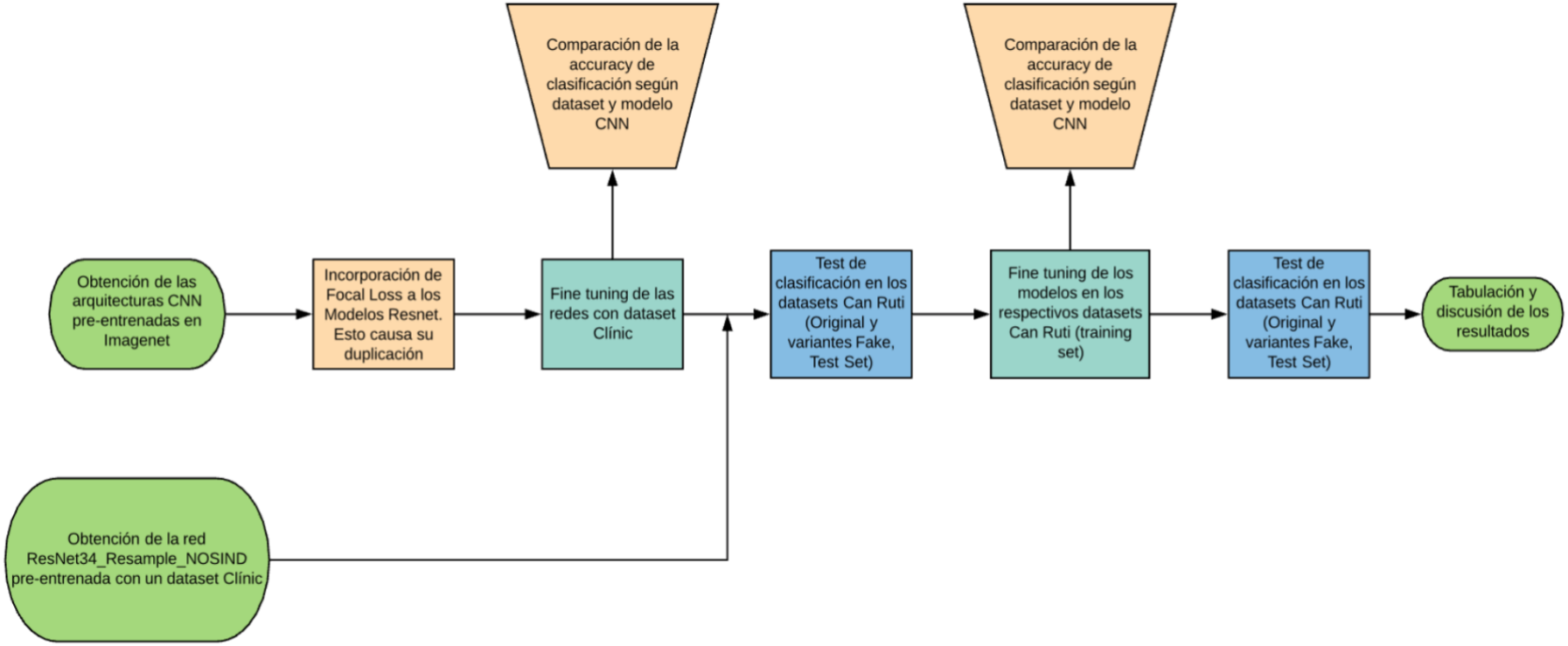


Figura 6.8. Diagrama del bloque de clasificación automática de imágenes de Can Ruti. Fuente: Elaboración Propia.

6.2.1. Fine tuning en el conjunto de muestreo Clínic

6.2.1.1. Redes utilizadas y *Focal Loss*

Las CNNs que se han utilizado se enumeran a continuación:

- *ResNet 18*
- *ResNet 18 Focal Loss*
- *ResNet 34*
- *Resnet 34 Focal Loss*
- *ResNet_34_Resample_NoSIND*

Se ha decidido implementar una *Focal Loss* [81] (ver Fragmento de código 6) que según la literatura ayuda a converger los modelos cuando se entrena con clases desbalanceadas como es este caso (véase la Figura 6.9 para la explicación matemática).

```
1.  class FocalLoss(nn.Module): # Se hereda la superclase nn.Module
2.      def __init__(self, gamma=2):
3.  # introducimos el parámetro gamma (valor por defecto = 2)
4.          super().__init__()
5.          self.gamma = gamma
6.
7.      def forward(self, logit, target):
8.          target = target.float()
9.          max_val = (-
    logit).clamp(min=0) # se introducen los logits por una función ReLU
10.         loss = logit - logit * target + max_val + \
11.                ((-max_val).exp() + (-
    logit - max_val).exp()).log() #se crea la función de pérdida
12.
13.         invprobs = F.logsigmoid(-logit * (target * 2.0 - 1.0))
14.         loss = (invprobs * self.gamma).exp() * loss
15.         if len(loss.size())==2:
16.             loss = loss.sum(dim=1)
17.         return loss.mean()
18.
19. learner_focal = None
20. learner_focal = cnn_learner(data, models.resnet34, metrics=[error_rate,accuracy
    ], pretrained='imagenet')
21. # Se carga el modelo, se incluyen las métricas
22. learner_focal.loss_fn = FocalLoss() # Se añade la Focal Loss
```

Fragmento de código 6. Implementación de la focal loss. Fuente: [64].

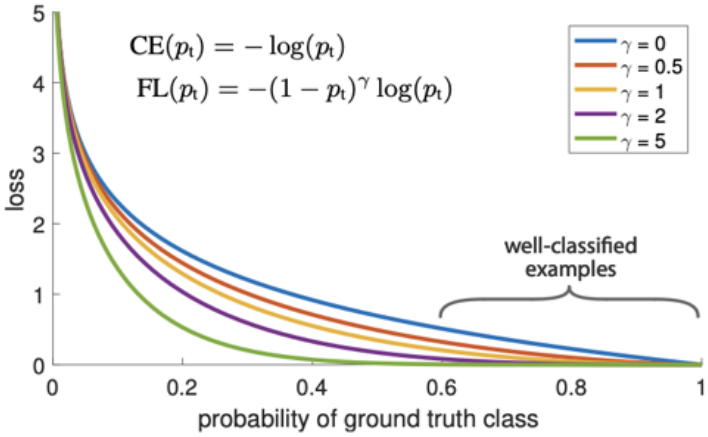


Figura 6.9. Imagen de la actuación de la focal loss (FL) en contraposición a la cross-entropy (CE). La FL surge de añadir el coeficiente (1 - p_t) a la CE. Obligando $\gamma > 0$ se reduce la pérdida relativa para los ejemplos bien clasificados ($p_t > .5$), poniendo más atención en los ejemplos más difíciles y mal clasificados. Fuente: [81].

6.2.1.2. Explicación del proceso de fine tuning en el conjunto de muestreo Clínic

Este entrenamiento se ha realizado con la ayuda del conjunto analizado en el Apartado 5.1. (Tabla 3). Los ensayos son consultables en el repositorio de *Github* (directorio **/Notebooks_Results_TFG/Experiments_3_Classes**). Se han entrenado varios *epochs* con el *training set* y se ha realizado una validación en el *testing set* para saber cuan bueno es el modelo clasificando imágenes que nunca ha visto.

En primer lugar, se importan los datos de la carpeta **folder_classes** (no visible en el *Github* por motivos de privacidad), donde se han movido las imágenes de las tres clases a clasificar mediante scripts de *Python* como los del Fragmento de código 1. Dentro de esta carpeta hay dos carpetas de **train** *y* **test**, dentro de las cuales hay tres carpetas que contienen imágenes (una para cada clase).

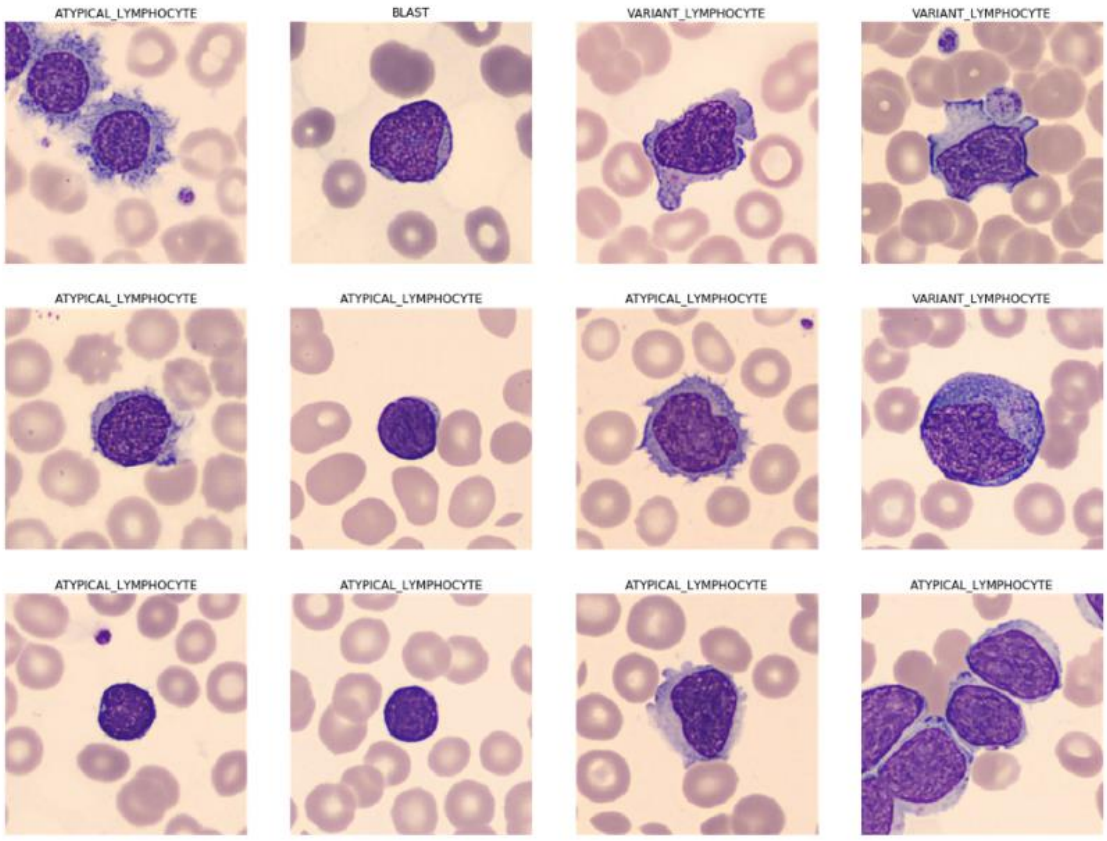
Se muestra a continuación la importación de las imágenes en la clase **Databunch** (que hace posible la introducción de datos en la red) de la librería **Fastai** por medio del **Datablock API** (módulo de dicha librería):

```
1.  folder_classes = "Directorio donde se encuentran las imágenes del conjunto
    Clínic"
2.  tfms = get_transforms(flip_vert=True, max_warp=None, max_zoom=1.01, max_rotate=
    120, max_lighting=0.1)
3.  data = (ImageList.from_folder(folder_classes)
4.  # Directorio de las carpetas train y test
5.          .split_by_rand_pct(seed=4)
6.  # Separamos los sets de train y validation en un ratio (80/20%)
7.          .label_from_folder()
8.  # Nombramos las clases mediante el nombre de las carpetas
9.          .transform(tfms, size=224)
10. # Data augmentation especificado en la variable tmfs
11.         .databunch()
```

```
12.          .normalize(imagenet_stats))
```

Fragmento de código 7. En las dos primeras líneas se insta el objeto que alberga las transformaciones de la data augmentation, no se incluye warp y se varía solo un 0.1 puntos la iluminación. Se posibilita el zoom y rotaciones. Posteriormente cargamos los datasets y normalizamos según el dataset imagenet, ya que las redes se han preentrenado con él.

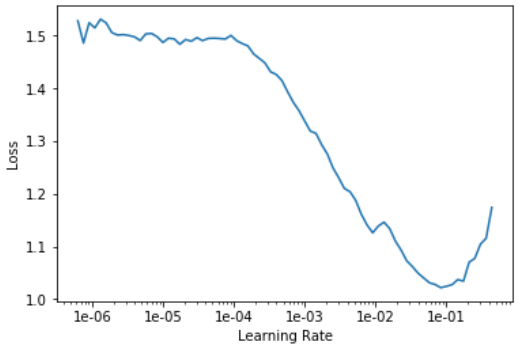
```
1.  data.show_batch()
```



Fragmento de código 8. Se muestra mediante el método `show_batch` un mini-batch (output) de la variable databunch.

Posteriormente se carga el modelo (en el Fragmento código 9 se muestra la importación *ResNet18*) y se incluyen las métricas de *error_rate* y *accuracy* (siempre suman 1). Esto se hace en la instancia del objeto **`cnn_learner`**, al que se le atribuyen los métodos y funciones para entrenar, testear y visualizar resultados. Se intenta visualizar una *learning rate* favorable para el entrenamiento, como se puede ver en el Fragmento de código 9. Aunque haya muchos métodos sobre como seleccionarlo, se ha optado por la política de un ciclo [82].

```
1. learn = cnn_learner(data, models.resnet18, metrics=[error_rate, accuracy], pret
   rained='imagenet')
2. learn.data.batch_size = 20
3. learn.lr_find()
4. learn.recorder.plot()
```



Fragmento de código 9. Creación del learner y descarga de la ResNet, se introducen las métricas y se busca un lr mediante el método `lr_find` (es aconsejable seleccionar un punto inmediatamente anterior a un cambio drástico). También se explicita el tamaño del `batchsize`.

Se procede al entrenamiento de la red durante tantos *epochs* como se crea necesario para lograr la convergencia del modelo (basándose los resultados de *accuracy* sobre el *validation set*). Se intentará que la **`valid_loss`** no supere de forma excesiva la **`training_loss`**, ya que eso sería una muestra de *overfitting*.

```
1. learn.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.551613	0.241443	0.109966	0.890034	00:10
1	0.396501	0.197874	0.079038	0.920962	00:11
2	0.258137	0.137352	0.049828	0.950172	00:10
3	0.206484	0.122323	0.051546	0.948454	00:10
4	0.189221	0.120455	0.041237	0.958763	00:11

```
2. learn.unfreeze()
3. learn.fit_one_cycle((5), max_lr=slice(
   3-e4,3-3e2))
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.363322	0.365199	0.092784	0.907216	00:11
1	0.469639	1.419887	0.250859	0.749141	00:11
2	0.325330	0.103184	0.037801	0.962199	00:11
3	0.213607	0.085662	0.022337	0.977663	00:11
4	0.116765	0.068961	0.017182	0.982818	00:11

Fragmento de código 10. Se muestran partes del entrenamiento (no todo). Al principio se entrenan varios epochs solo las dos últimas capas de la red a modo de "inicialización de pesos". Posteriormente se descongela toda la red mediante el método "unfreeze" y se entrenan todas las capas descongeladas variando el learning rate.

Cuando se ha acabado con el entrenamiento se guarda el modelo con el método **`export("Nombre del modelo")`**. De esta forma se exporta la arquitectura con los pesos aprendidos hasta el momento dentro de la carpeta del *dataset* en uso.

```
1. learn.export('export_stage-1-18-Clinic_Balanced_3_Classes')
```

Fragmento de código 11. Ejemplo de exportación del modelo de CNN.

Finalmente se realiza la validación del *testing set* del conjunto Clínic como se muestra en el Fragmento de código 12. Se puede observar, como ejemplo, la matriz de confusión de la clasificación del *testing set* con la *ResNet 18* en la Tabla 8. En la Tabla 9 se recapitulan los las clasificaciones en el *conjunto testing* con todas las CNN.

```
data_test = (ImageList.from_folder(folder_classes)
# Directorio de las carpetas train y test
        .split_by_folder(train='train', valid='test')
# Se selecciona la carpeta de test para validar
        .label_from_folder() # Etiquetación de la imagen según carpeta
        .transform(tfms, size=224)
        .databunch()
        .normalize(imagenet_stats))

metrics_valid = learn_test.validate(data_test.valid_dl)

interp = ClassificationInterpretation.from_learner(learn)
# Instancia del objeto que contiene la matriz de confusión
interp.plot_confusion_matrix(figsize=(6,6), dpi=110)
 # Plot de la matriz de confusión
```

Fragmento de código 12. Carga del testing set y validación de este, posterior extracción de la matriz de confusión.

Tabla 8. Matriz de confusión normalizada resultante de la clasificación del conjunto testing con la ResNet 18.

Confusion Matrix			
Actual: Atypical Lymphocyte	0.92	0.08	0.00
Actual: Blast	0.00	1.00	0.00
Actual: Variant Lymphocyte	0.00	0.03	0.97
	Predicted: Atypical Lymphocyte	Predicted: Blast	Predicted: Variant Lymphocyte

Tabla 9. Resultado (accuracy del modelo) de las clasificaciones con todas las CNN en el conjunto Clínic.

Modelo de CNN	Accuracy en el testing set del Conjunto Clínic post *fine tuning*
ResNet_18	0.981061
ResNet_18_Focal	0.981061
Resnet_34	0.981061
ResNet_34_Focal	0.965909
ResNet_34_No_Sind (Solo validación en el conjunto testing)	0.984848

Como se puede observar el *fine tuning* es exitoso en el conjunto *testing* Clínic para todos los modelos.

6.2.2. Clasificación en los datasets Can Ruti (originales y fake)

Estos ensayos se realizarán con las CNNs modificadas anteriormente (tras el *fine tuning* sobre el conjunto Clínic del punto anterior) en el *set* de validación de las siguientes variantes de conjuntos Can Ruti:

- Can Ruti Original
- Can Ruti *Fake CycleGAN* 15 *epochs (No* Dataset *Separate)*
- Can Ruti *Fake CycleGAN* 20 *epochs (No* Dataset *Separate)*
- Can Ruti *Fake CycleGAN* 200 *epochs (No* Dataset *Separate)*
- Can Ruti *Fake CycleGAN* (Dataset *Separate*) e*pochs* óptimos
- Can Ruti *Fake CycleGAN* (Dataset *Separate*) 200 *epochs*

El modelo y los *datasets* se cargan de forma similar a la explicada en el apartado anterior, para cada *dataset* diferente se introduce su directorio en el código de *Datablock API.* Esto es porque dichas imágenes están organizadas en carpetas diferentes dependiendo de cada variante de Can Ruti mencionada en la lista anterior (véase el Fragmento de código 19 del Anexo para ver como se crean los *dataroots* de las variantes generadas del conjunto Can Ruti). Para cargar los modelos que han realizado *fine tuning* en Clínic se utiliza la siguiente función:

```
1. learn = load_learner(folder_model, 'export_stage-2-18-
   Clinic_Balanced_3_Classes')
2. learn.data = data
```

Fragmento de código 13. Se utiliza la función `Load_Learner` para cargar el modelo en la variable "learn". En la segunda línea se introduce el `databunch` a clasificar.

Para realizar el test de clasificación se utiliza el método **`validate`** introduciéndole el objeto **`data.valid_dl`**. Este atributo (**`dataloader`** del *validation set*) posibilita la carga de los datos en el modelo para realizar clasificación.

```
1. results = learn.validate(data.valid_dl)
```

Fragmento de código 6. Validación del propio validation set de una variante del conjunto Can Ruti.

Cabe destacar que la métrica extraída de precisión ha sido obtenida mediante la elaboración de una función gracias a la librería **`sklearn`** (Fragmento de código 15). Como esta métrica está restringida a una clasificación binaria se ha optado por obtener las etiquetas de predicción y las reales del conjunto de validación y binarizarlas. Se ha aplicado la técnica de "micro-promediado" en las funciones (utilización del argumento **`average="micro"`**) para así tener en cuenta el desbalance de las clases.

Para realizar una separación de *training* y *validation set* en todas las variantes del conjunto Can Ruti se ha utilizado una rutina especificada en el Fragmento de código 20 (ver Anexo A.4).

```
plt.rcParams.update({'font.size': 18}) # Importamos las librerías necesarias
from sklearn.metrics import precision_score

val_preds,val_targets = learn.get_preds()

y_test = pd.Series(np.array(val_preds.max(1)[1])).map(dict(zip([0,1,2],learn.data.classes)))
y_true = pd.Series(np.array(val_targets)).map(dict(zip([0,1,2],learn.data.classes)))
# Cambiamos las etiquetas 1-hot-encoded por variables categóricas

precision = precision_score(y_test, y_true, average='micro')
```

Fragmento de código 7. Obtención de la métrica de precisión en un problema multiclase.

6.2.2.1. Resultados de las clasificaciones en los datasets Can Ruti (originales y *fake*)

El resultado de los test se ha tabulado a continuación, las tablas siguientes reflejan en este orden *accuracy,* matrices de confusión y precisión.

Los resultados en rojo (Tabla 10) son los que no han superado el 0.5 de *accuracy*, representan un pobre desempeño del modelo clasificador (es el ejemplo de las ejecuciones en el *dataset* Can Ruti original y en el resultante de la *ColorizationGAN*). Las *accuracys* en verde son las que han superado el *0.6.* En el caso de la transformación previa con *CycleGAN* (a 15 y 20 *epochs*) se demuestra una mejora considerable del desempeño del modelo con respecto a la clasificación en el *dataset* original.

Tabla 10 .Tabla de los accuracies en la clasificación de las distintas variantes del conjunto Can Ruti antes del fine tuning.

Dataset de Clasificación	Can Ruti Original	Can Ruti CycleGAN Dataset No Separate			Can Ruti CycleGAN Dataset Separate		Can Ruti Colorization GAN
Modelo de CNN		Can Ruti Fake 15 Epochs	Can Ruti *Fake 20 Epochs*	Can Ruti *Fake 200 Epochs*	Can Ruti *Fake Separate Epochs Óptimos*	Can Ruti *Fake Separate 200 Epochs*	Can Ruti Fake Colorization 200 Epochs
ResNet_18	0.3152	0.5869	0.5326	0.4782	0.6630	0.9565	0.3152
ResNet_18_Focal	0.4565	0.6847	0.5869	0.5326	0.8043	0.9673	0.3804
Resnet_34	0.2826	0.5543	0.5326	0.5	0.7826	1	0.3260
ResNet_34_Focal	0.3043	0.6739	0.5326	0.5326	0.8804	0.9891	0.3586
ResNet_34_No_Sind	0.3478	0.8152	0.75	0.5326	0.8587	0.9891	0.5108

Por lo que refiere a la clasificación del *dataset* Can Ruti *Fake* a 200 *epochs* (CycleGAN), éste presenta una mejora respecto a la clasificación con el *dataset* original en todos los modelos, aunque debido a los

cambios morfológicos que produce se ha decidido no continuar con ella en el posterior paso de *fine tuning* en el mismo *dataset*. Lo mismo aplica para la clasificación del *Dataset Separate Fake* (Can Ruti) a 200 *epochs* (*CycleGAN),* que ha conseguido un alto *accuracy* pero los cambios morfológicos son inadmisibles para esta aplicación. Cabe remarcar que la separación por clases ha causado un sesgo enorme en la transformación (sobretodo a 200 *epochs*) porque se está forzando una conversión morfológica a células del conjunto Clínic (más reconocibles para las CNN pre-entrenadas con *datasets* Clínic).

Los resultados en azul reflejan esta última idea: buenos resultados, pero con cambios morfológicos en las células. Esto es algo imposible de realizar en el caso de que el modelo esté en producción en una rutina de laboratorio. Esto es porque se desconocería a que categoría pertenece cada imagen a priori, ya que todavía no se habría realizado la clasificación y no se conocerían las etiquetas.

Se muestra en la Tabla 11 y la Tabla 12 las matrices de confusión del mejor resultado en cuanto a testeo sin cambios morfológicos evidentes. Éste el test con la ResNet_34_No_Sind en el *dataset* Can Ruti *Fake CycleGAN* (15 *epochs*). Se presentan más matrices de confusión en el Anexo (B.2).

Tabla 11. Matriz de confusión absoluta del mejor resultado en cuanto a testeo sin cambios morfológicos evidentes, test con la ResNet_34_No_Sind en el dataset Can Ruti Fake CycleGAN (15 epochs).

	Confusion Matrix		
Atypical Lymphocyte	1.00	0.00	0.00
Blast	0.00	1.00	0.00
Variant Lymphocyte	0.47	0.10	0.43
	Atypical Lymphocyte	Blast	Variant Lymphocyte

Predicted

Tabla 12. Matriz de confusión normalizada del mejor resultado en cuanto a testeo sin cambios morfológicos evidentes, test con la ResNet_34_No_Sind en el dataset Can Ruti Fake CycleGAN (15 epochs).

	Confusion Matrix		
Atypical Lymphocyte	25	0	0
Blast	0	37	0
Variant Lymphocyte	14	3	13
	Atypical Lymphocyte	Blast	Variant Lymphocyte

Predicted

Es digno de mención que la introducción de *focal loss* en los modelos *ResNet* ha mejorado considerablemente el desempeño respecto a las mismas redes que entrenan con la convencional *cross-entropy loss* en varias ocasiones. Esto puede ser debido al desbalance de clases en el entrenamiento, cosa que esta función de pérdida contempla y le presta especial atención. Asimismo, la ResNet34_No_Sind resulta ser bastante fiable, teniendo en cuenta que ha sido entrenada en un *dataset* considerablemente mayor, por lo que tendría menos dificultades para generalizar en conjuntos datos ajenos a los de entrenamiento.

A pesar de que el *Dataset Separate* a *epochs* óptimos ha sufrido cambios morfológicos no deseados, esta prueba y la realizada con el *dataset* no separado por clases a 15 *epochs* muestran resultados de mucho valor para esta investigación. Esto es que con una conversión previa a la clasificación que emule lo suficientemente bien las características del conjunto Clínic sin pervertir morfológicamente las imágenes ayuda enormemente aumentar la *accuracy* sin necesidad de hacer un *fine tuning* en el conjunto del hospital ajeno.

Para finalizar este punto, cabe mencionar que las clasificaciones generalmente han mostrado una alta sensibilidad para linfocitos atípicos y blastos. No obstante, los modelos se han mostrado poco sensibles para linfocitos variantes. Esto puede ser debido al desbalance de clases (ya que se cuenta con un número menor de ejemplares de esta categoría tanto en el entrenamiento de GANs como de CNN).

Tabla 13. Tabla de la precisión en la clasificación de las distintas variantes del conjunto Can Ruti antes del fine tuning.

Dataset de Clasificación	**Can Ruti Original**	**Can Ruti CycleGAN Dataset No Separate**			**Can Ruti CycleGAN Dataset Separate**		**Can Ruti Colorization GAN**
Modelo de CNN		**Can Ruti Fake 15 Epochs**	**Can Ruti *Fake 20 Epochs***	**Can Ruti *Fake 200 Epochs***	**Can Ruti *Fake Separate Epochs Óptimos***	**Can Ruti *Fake Separate 200 Epochs***	**Can Ruti Fake Colorization 200 Epochs**
ResNet_18	0.3152	0.5869	0.5326	0.4782	0.6630	0.9565	0.3152
ResNet_18_Focal	0.2826	0.6847	0.5869	0.5326	0.8043	0.9673	0.3804
Resnet_34	0.4565	0.5543	0.5326	0.5000	0.7826	1.0	0.3260
ResNet_34_Focal	0.3043	0.6739	0.5326	0.5326	0.8804	0.9891	0.3586
ResNet_34_No_Sind	0.3478	0.8152	0.75	0.5326	0.8586	0.9891	0.5108

6.2.2.2. Fine tuning de los modelos en los datasets (originales y fake) y nuevo test de clasificación

Después de las clasificaciones del apartado anterior se realizó el *fine tuning* en la mitad del dataset de Can Ruti (conjunto *training*) con el mismo criterio que en el Apartado 6.2.1.2, llegar a una buena *accuracy* del *validation set* sin que la **`validation_loss`** exceda en gran medida la **`train_loss`**. Para este proceso se ha contado con los modelos ya mencionados y los siguientes *datasets:*

- Can Ruti Original
- Can Ruti *Fake CycleGAN* 15 *epochs*
- Can Ruti *Fake CycleGAN* 20 *epochs*
- Can Ruti *Fake CycleGAN Dataset Separate* e*pochs* óptimos
- Can Ruti *Fake ColorizationGAN* 200 *epochs*

Los *datasets* que faltan se han excluido por las transformaciones morfológicas sufridas en el proceso de transformación, ya que se ha considerado que no aportaban información útil para la aplicación del *fine tuning* ni para la posterior prueba de clasificación.

6.2.2.3. Resultados del nuevo test de clasificación en el conjunto Can Ruti (después del fine tuning)

Como se puede observar en la Tabla 14, casi todos los resultados superan el 90% de *accuracy*. Las pruebas que tienen los desempeños más altos son las de los modelos empleados en el conjunto Can Ruti *Fake CycleGAN* (*Dataset Separate)* de e*pochs* óptimos. Esto puede ser engañoso debido al entrenamiento que tuvo la *CycleGAN* separado por clases para realizar la conversión. Por ello no se considerará como el mejor resultado.

Sorprendentemente, las clasificaciones realizadas del *dataset* Can Ruti original han aumentado considerablemente el *accuracy* tras el *fine tuning* con el *training set* de este conjunto*,* mientras que las clasificaciones de *datasets* resultantes de las transformaciones de la *CycleGan (no Dataset Separate)* no llegan a cotas tan altas de desempeño.

El *dataset* Can Ruti *Fake ColorizationGAN* (200 *epochs*) también ha aumentado su *accuracy* de clasificación de gran manera. Por otra parte, hay que mencionar que la *focal loss* (en algunas ocasiones) ha mejorado el desempeño respecto a las mismas redes con la convencional *cross-entropy loss* incorporada.

Por lo general, al haber conseguido estas cotas de *accuracy* tan altas en el conjunto Can Ruti, cabría pensar que en un caso real podría evitarse el preprocesado por conversión mediante *GAN.* No obstante, no siempre se cuenta con una cantidad suficiente de datos de hospitales ajenos para realizar el *fine*

tuning en esa distribución de datos, por lo que la disponibilidad *CycleGAN* para realizar la conversión podría servir para pre-procesar las imágenes y clasificar con buenos resultados.

Tabla 14. Tabla de accuracies en la clasificación de las distintas variantes del conjunto Can Ruti usando los modelos de CNN posteriores al fine tuning de este apartado.

Dataset de Clasificación	Can Ruti Original	Can Ruti CycleGAN Dataset No Separate		Can Ruti CycleGAN Dataset Separate	Can Ruti Colorization GAN
Modelo de CNN		Can Ruti Fake 15 Epochs	Can Ruti *Fake 20 Epochs*	Can Ruti *Fake Separate Epochs Óptimos*	Can Ruti Fake Colorization 200 Epochs
ResNet_18	0.9565	0.9130	0.8804	1	0.9565
ResNet_18_Focal	0.9673	0.9782	0.9239	1	0.9782
Resnet_34	0.9673	0.9673	0.9239	1	0.9782
ResNet_34_Focal	0.9239	0.9347	0.8913	1	0.9347
ResNet_34_No_Sind	0.9891	0.9239	0.8913	1	0.9673

Tabla 15. Número de epochs de entrenamiento de la clasificación de las distintas variantes del conjunto Can Ruti usando los modelos de CNN posteriores al fine tuning de este apartado.

Dataset de Clasificación	Can Ruti Original	Can Ruti CycleGAN Dataset No Separate		Can Ruti CycleGAN Dataset Separate	Can Ruti Colorization GAN
Modelo de CNN		Can Ruti Fake 15 Epochs	Can Ruti *Fake 20 Epochs*	Can Ruti *Fake Separate Epochs Óptimos*	Can Ruti Fake Colorization 200 Epochs
ResNet_18	9	19	30	7	20
ResNet_18_Focal	7	20	23	2	13
Resnet_34	37	15	18	6	12
ResNet_34_Focal	19	16	18	9	17
ResNet_34_No_Sind	9	23	17	5	18

Tabla 16. Matriz de confusión del test con la ResNet_34_No_Sind en el dataset Can Ruti Original después del fine tuning en éste.

Confusion Matrix			
Atypical Lymphocyte	0.96	0.00	0.04
Blast	0.00	1.00	0.00
Variant Lymphocyte	0.00	0.00	1.00
	Atypical Lymphocyte	Blast	Variant Lymphocyte

Actual

Predicted

Tabla 17. Matriz de confusión normalizada (derecha) del test con la ResNet_34_No_Sind en el dataset Can Ruti Original después del fine tuning en éste.

Confusion Matrix			
Atypical Lymphocyte	24	0	1
Blast	0	37	0
Variant Lymphocyte	0	0	30
	Atypical Lymphocyte	Blast	Variant Lymphocyte

Actual

Predicted

6.3. Bloque de transformación de imágenes de linfocitos normales a células de sangre periférica anómalas y viceversa (conjunto Clínic)

La manera de proceder de este bloque se resume en el diagrama de la Figura 6.10. Los pasos a seguir serán los siguientes:

1. Organización de las imágenes en diferentes *datasets* de *entrenamiento* y *testing* de la *CycleGAN* (ver Apartado 5.4).
2. Selección del *preset* de los parámetros de entrenamiento de la *CycleGAN* y consiguientes entrenamientos con cada uno de los *datasets.*
3. Valoración cualitativa de las transformaciones realizadas en ambas direcciones y comparación de estas con la *ground truth* (imágenes verdaderas) de la clase de célula a la que se ha transformado.

6.3.1. Criterios de formación de los *datasets* para los entrenamientos del Bloque 3

Para crear los datasets de entrenamiento de la CycleGAN para la aplicación de este bloque se obrará de forma análoga a lo ya explicado en el Apartado 6.1.1. Por otra parte, cabe destacar que el criterio de selección para las clases enfrentadas en estos ensayos ha sido la cantidad de imágenes disponibles. Los entrenamientos de ejemplo en el repositorio de *Pytorch* del proyecto *CycleGAN* [80] contienen al menos 1000 imágenes para cada clase.

6.3.2. *Preset* de parámetros para los entrenamientos del Bloque 3

El *preset* de entrenamiento y *testing* para las transformaciones en este bloque ha sido idéntico al del Apartado 6.1.2.1, ya que la aplicación es muy similar. No obstante, se ha finalizado el entrenamiento en 200 *epochs* para que la transformación sea lo suficientemente elaborada.

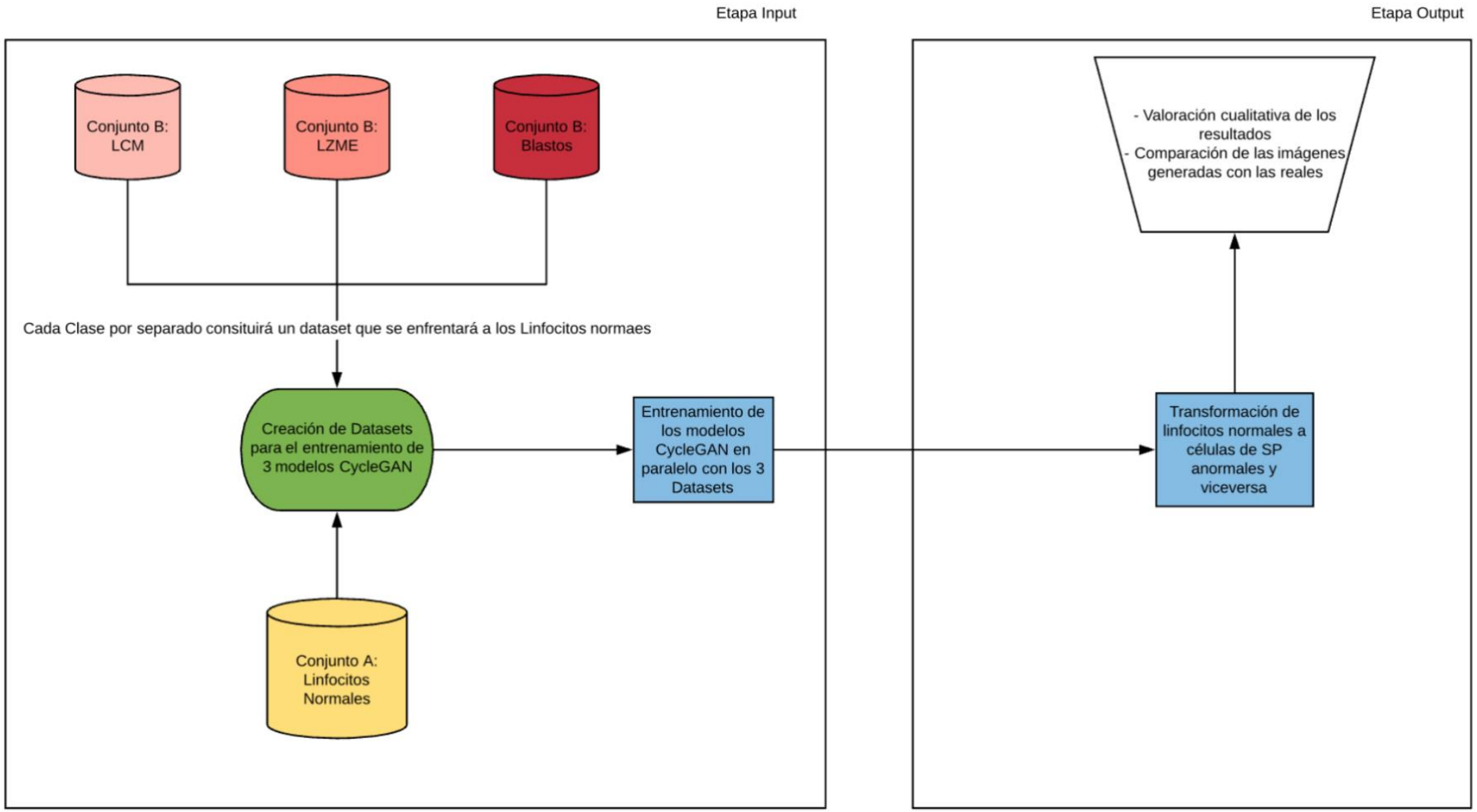


Figura 6.10. Diagrama del bloque de transformación de imágenes de linfocitos normales a células de sangre periférica anómalas y viceversa (conjunto Clínic). Fuente: Elaboración Propia.

6.3.3. Discusión de los resultados del Bloque 3

Las transformaciones de ejemplo de las ilustraciones de este apartado son consultables en el repositorio de *Github* (directorio **`/Notebooks_Results_TFG/results_TFG/Bloque_3`**).

6.3.3.1. Transformación de linfocitos normales vs. linfocitos LZME (y viceversa)

La ejecución de este experimento, visible en la Figura 6.11, ha resultado satisfactoria. Como se puede observar, se ha aumentado el citoplasma alrededor del núcleo en la transformación de normal en LZME (imitando la textura pilosa). En la transformación inversa se ha reducido este citoplasma y oscurecido la cromatina del núcleo como es común en muchos linfocitos normales.

Algunas veces se altera la morfología de los eritrocitos de alrededor. También hay que destacar que en algunas células se conserva un resquicio del contorno del núcleo original.

El cambio del color y las texturas de las imágenes *fake* se equiparan bastante bien a las características de la clase de célula que se quiere conseguir en ambos sentidos de la transformación.

6.3.3.2. Transformación de linfocitos normales vs. blastos (y viceversa)

En cuanto a este ensayo, cabe subrayar que al igual que en el anterior se han conseguido transformaciones convincentes, véase la Figura 6.12.

Nótese que en algunos casos la transformación del linfocito no es del todo perfecta, pero sorprendentemente se ha mejorado la forma de los eritrocitos (como el caso de la 1ª y 2ª columna, 3ª fila). También desaparece algún que otro eritrocito (ejemplo: 4ª fila)

En cuanto a transformaciones de color y textura, se consigue producir colores de cromatina más oscuros en el caso de los linfocitos normales *fake* y más claros en los blastos *fake.* Por otro lado, la variación del tamaño del núcleo como cambio morfológico semántico de alta importancia ha sido bien asimilado por la *CycleGAN* (aumentándolo para generar blastos *fake* y disminuyéndolo para generar linfocitos normales *fake*).

6.3.3.3. Transformación de linfocitos normales vs. linfocitos LCM

Este experimento ha sido más dificultoso para la *CycleGAN*, ya que se han introducido células LCM de tres tipos distintos (indolente, blástico y típico). Como se muestra en la 1ª y 2ª columna de la fila 3ª (Figura 6.13) se han transformado eritrocitos en LCM. En la transformación de linfocitos normales en LCM se ha aumentado el núcleo y disminuido en la transformación inversa, añadiendo citoplasma de pequeño grosor. Se puede concluir en que la transformación de LCM en normales ha sido de mayor

calidad, puesto que la heterogeneidad del *input* de los LCM ha podido confundir a la red transformadora.

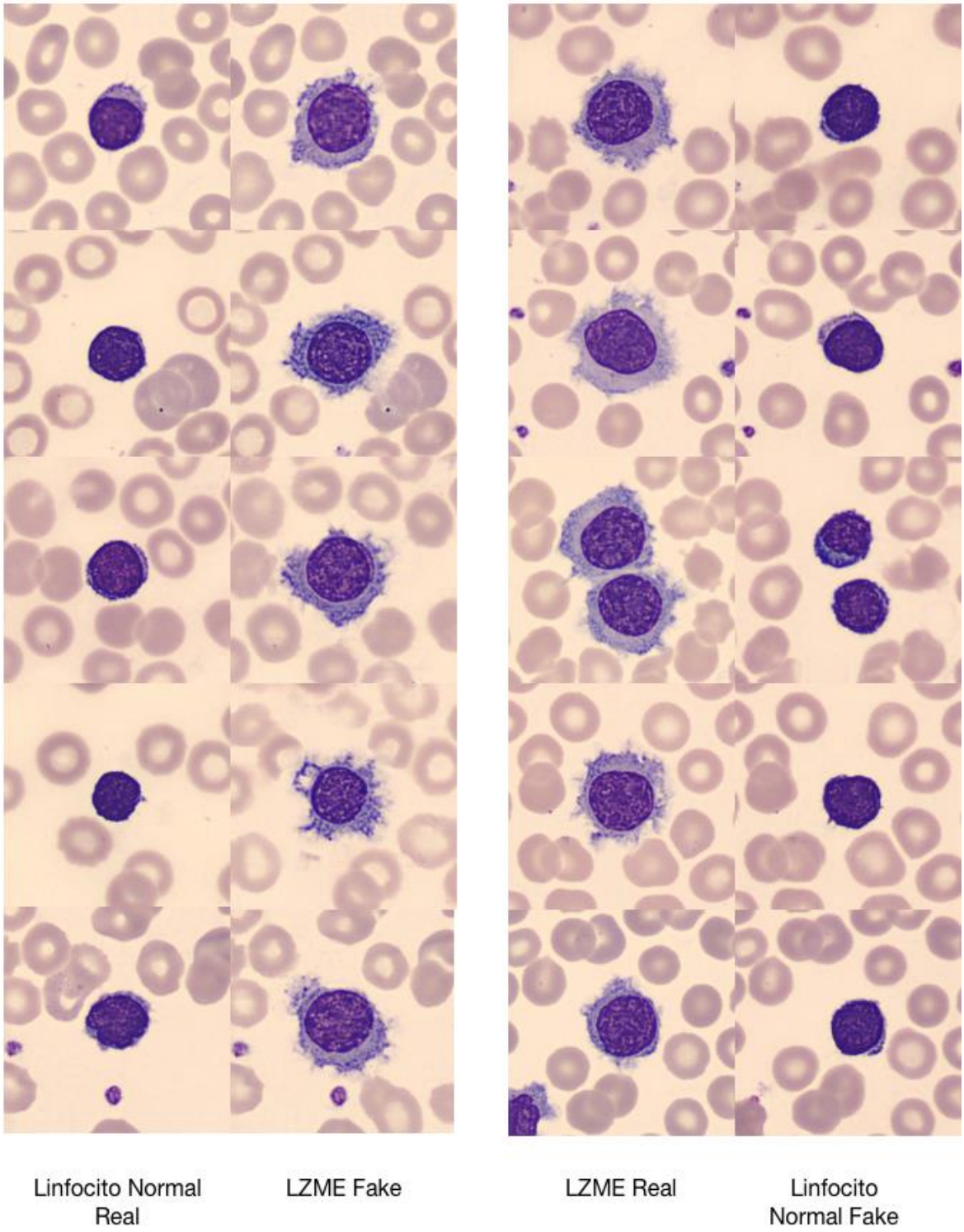


Figura 6.11. Resultados de la transformación Linfocito Normal vs LZME (y viceversa). Fuente: Imágenes del Datast Clínic y generadas.

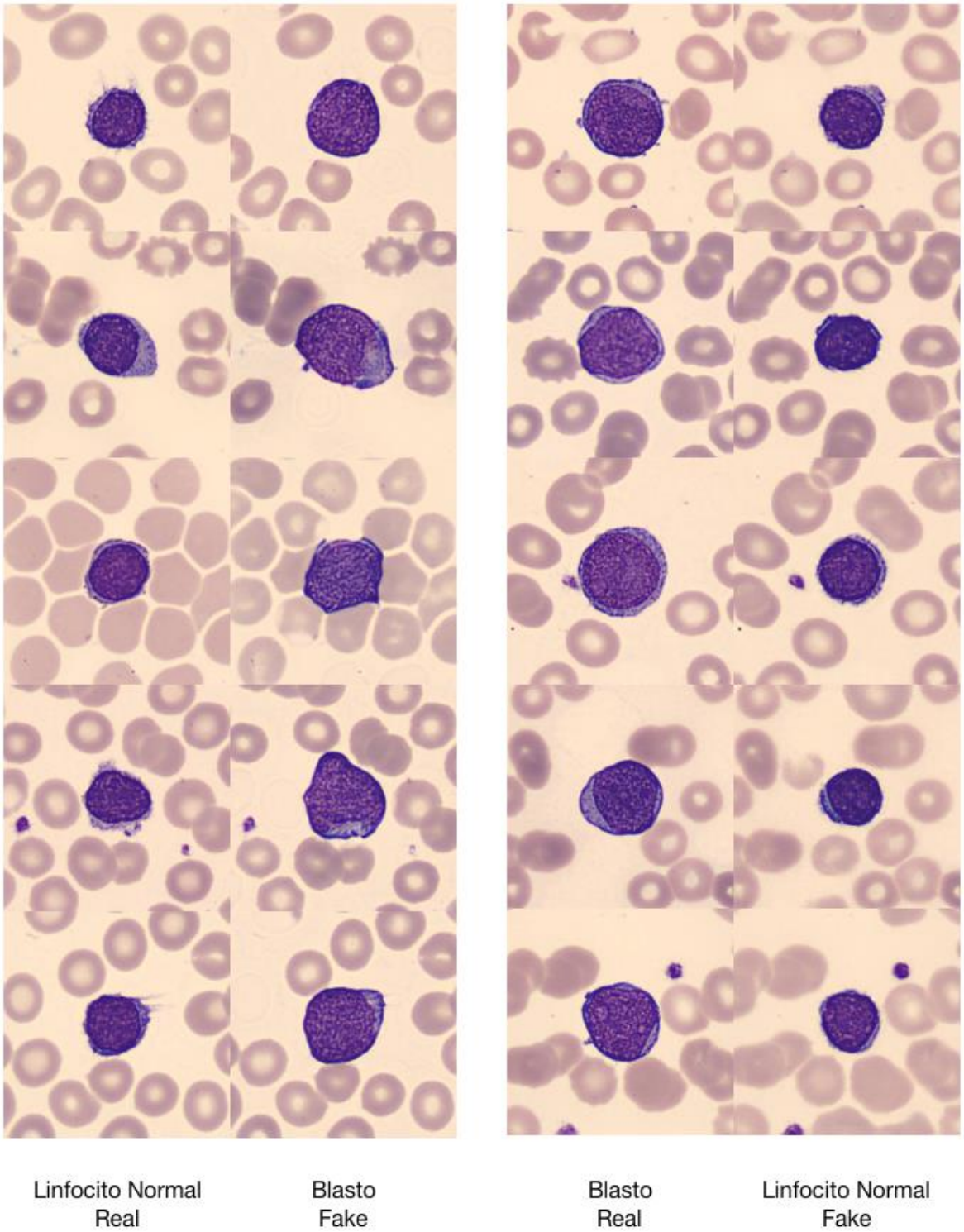


Figura 6.12. Resultados de la transformación Linfocito Normal vs Blastos (y viceversa). Fuente: Imágenes del Datast Clínic y generadas.

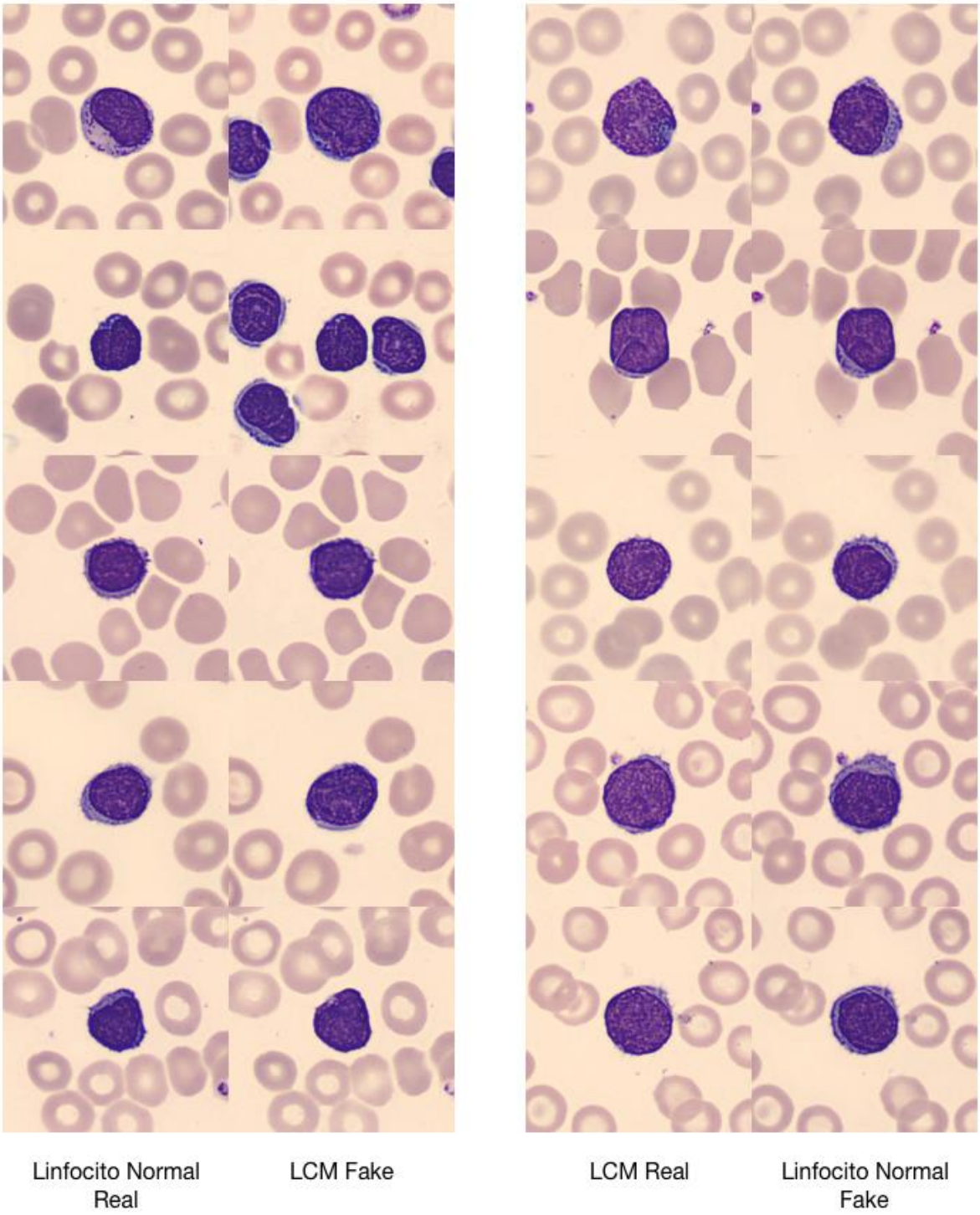


Figura 6.13. Resultados de la transformación Linfocito Normal vs LCM (y viceversa). Fuente: Imágenes del Datast Clínic y generadas.

7. Análisis de impacto ambiental

En cuanto a impacto en el diagnóstico de enfermedades en un hipotético uso de los modelos GAN en producción, hay que comentar que el pre-procesado de imágenes de hospitales ajenos al Clínic a modo de transformación podría ahorrar tiempo de ejecución. Es decir, si se prefiere, podría utilizarse un modelo *CycleGAN* con el fin de emular las características de los *datasets* del hospital local para así evitar calibrar (hacer *fine tuning*) de nuevo. El *fine tuning* o la creación de otros modelos para la clasificación de células de otros hospitales genera un retraso de tiempo en el diagnóstico. Además, se necesita entrenar con grandes cantidades de imágenes de células para que un modelo sea lo suficientemente sensible y competente para clasificar todas las clases de linfocitos.

Cabe destacar que, por ahora, el estado del arte de esta tecnología limita estos clasificadores a un uso auxiliar, ya que la decisión final en el diagnóstico y la praxis médica la tiene el equipo de especialistas médicos. No obstante, son de gran ayuda para hacer pre-clasificaciones células de frotis como una herramienta más de los equipos técnicos y así saber qué otros estudios (oncológicos, genéticos, de imagen médica, etc.) realizar sobre del paciente.

Si un modelo transformador *GAN* llegase a producción, se aconseja la periódica comprobación de la calidad de sus transformaciones en base a la exactitud de modelos clasificadores mediante el test en *sets* de validación. También sería ideal comprobar a cuántos *epochs* hay que entrenar la red GAN para realizar conversiones que no perviertan la morfología celular.

Por último, cabe destacar que, si algún componente *hardware* como los servidores físicos u ordenadores se averíen o se les acabe la vida útil, se insta desecharlo en un punto verde de forma debida. Algunos de estos dispositivos están compuestos de plásticos no degradables o metales pesados que pueden contaminar el medio ambiente. Por ello se debe seguir la *Directiva 2012/19/UE* [83] que regla la retirada de residuos. Su homologación en España se el *Real Decreto 110/2015*.

8. Conclusiones y perspectivas futuras

8.1. Conclusiones

En lo que respecta al primer bloque de la metodología de este TFE, el principal objetivo ha sido la transformación de imágenes de células de sangre periférica utilizando Redes Neuronales Generativas Antagónicas. Las conclusiones derivadas del estudio con respecto al primer objetivo son las siguientes:

1. Se ha conseguido realizar una conversión de imágenes de células de sangre periférica pertenecientes al Hospital Germans Trias i Pujol (Hospital Can Ruti) emulando los estándares de color, tinción y textura de las imágenes tomadas en el Hospital Clínic de una forma convincente. Esto ha sido posible gracias a la *CycleGAN*, la arquitectura que ha conseguido una transformación de mayor calidad en comparación a la otra utilizada, la *ColorizationGAN*.

2. Se ha podido conformar un algoritmo de preprocesado consistente en el entrenamiento de este modelo y posterior testeo (proceso de transformación de las imágenes) para ser clasificadas de forma automática en una etapa clasificadora. En un escenario real, contando con un modelo lo suficientemente bien entrenado para realizar conversiones de alta calidad, no haría falta volver a hacer *fine tuning* siempre, bastaría con solo utilizar el modelo en modo test para realizar la transformación de las imágenes

3. Se han podido valorar cualitativamente los distintos tipos de imágenes generadas (Can Ruti *Fake)* resultantes de la transformación mediante distintos modelos y distintas técnicas (variación del número de *epochs* de entrenamiento de la *CycleGAN* y utilización de diversas arquitecturas). Se ha podido comparar éstas con imágenes reales del conjunto Clínic y entre las mismas imágenes generadas por distintas técnicas.

Por medio del segundo bloque de metodología cuyo objetivo ha sido clasificar de forma automática las imágenes pertenecientes al Hospital Can Ruti (originales y generadas) utilizando redes CNN clasificadoras se han obtenido las siguientes conclusiones:

1. Se han podido clasificar de forma automática mediante CNNs las imágenes tomadas en el Hospital Can Ruti al igual que los diferentes conjuntos de imágenes transformadas. Esto se ha realizado gracias al fine tuning previo en el conjunto Clínic (salvo en la red ResNet34_NoSIND, que ya fue preentrenada en un *dataset* de imágenes tomadas en el Hospital Clínic de mayor tamaño).

2. Se ha podido comparar la *accuracy* de clasificación de los diferentes *datasets* Can Ruti (el original y los *fakes*). La *accuracy* de clasificación ha servido para determinar que la técnica de transformación de imágenes más efectiva es la *CycleGAN* entrenada 15 *epochs*. El modelo ResNet34_NoSIND ha conseguido una *accuracy* del 81,52% en este *dataset*, aumentando aproximadamente 60 puntos porcentuales de *accuracy* con respecto a la clasificación del conjunto Can Ruti original.

3. Se ha comprobado la alta efectividad del *fine tuning* si se realiza en conjuntos de entrenamiento de las variantes del *dataset* Can Ruti, tanto en las generadas como la original. Esto supone una alternativa al preprocesado mediante CycleGAN en caso que éste no esté disponible. Mediante este método se ha conseguido llegar a desempeños que superan el 90% con todos los modelos en el *dataset* original, llegando incluso al 98,91%.

En lo que refiere al tercer bloque del TFE, cuyo objetivo ha sido la utilización de GANs para transformar células sanguíneas normales a anormales o viceversa por medio del aprendizaje automático y la inteligencia artificial, se concluye lo siguiente:

- Ha sido posible la transformación de imágenes de células normales a otros tipos de células malignas gracias a la arquitectura *CycleGAN*. La conversión se ha realizado también a la inversa y ha sido satisfactoria generalmente en ambos sentidos. Por último, se ha podido realizar una comparación y valoración cualitativa de ambas transformaciones.

La posibilidad de transformar las imágenes de células sanguíneas de un tipo a otro abre la posibilidad del entrenamiento de clasificadores con imágenes de células malignas correspondientes a enfermedades poco prevalentes. La obtención de un número adecuado de imágenes que contengan este tipo de células de enfermedades poco prevalentes resulta muy difícil, por lo que el uso de la arquitectura *CycleGAN* para su obtención puede ser de gran utilidad.

8.2. Perspectivas futuras

Derivadas de los satisfactorios resultados del presente TFE las acciones futuras que se consideran de interés son las siguientes:

1. Ampliar la aplicación de las Redes Neuronales Generativas Antagónicas a imágenes obtenidas en otros hospitales del país con objeto de mejorar los resultados de clasificación automática con CNNs

entrenadas con imágenes del Clínic. La idea es poder clasificar automáticamente imágenes de células sanguíneas anormales independientemente de su procedencia y método de tinción.

2. Aumentar el número de tipos de células malignas en el entrenamiento de los clasificadores del Clínic, como por ejemplo la leucemia tipo Burkitt u otros grupos de los que se dispone de muy pocas imágenes, utilizando la arquitectura CycleGAN para la generación de imágenes artificiales de estas entidades.

3. Relacionado también con un aumento de tamaño de los datasets de entrenamiento, se podría aumentar el número de imágenes de células de las categorías empleadas en este TFE para que el proceso de entrenamiento-transformación se desarrolle con una mayor precisión y calidad.

4. Idear un sistema lo más automático posible para encontrar a que *epochs* de entrenamiento la *CycleGAN* realiza transformaciones de color y textura sin comprometer la morfología celular. Esto podría lograrse mediante la transformación de la arquitectura o la modificación de las funciones de pérdida que intervienen en el entrenamiento.

9. Análisis Económico

9.1. Material y *hardware*

En este análisis económico no se van a incluir los computadores personales utilizados ya que en un entorno hospitalario profesional estarían proporcionados por la institución. La mayoría de *hardware* usado en la realización de este TFE es de utilización básica (un simple ordenador portátil o PC de escritorio) salvo el servidor que proporciona la estación de trabajo *deepbox.* Es por eso que la ya comentada estación de trabajo *deepbox* con *GPU Titan XP de 12 GB* se contará como un gasto indispensable.

Otro gasto similar es el del microscopio electrónico que tomó las imágenes citológicas. Como pertenecen a hospitales diferentes y se desconocen los modelos exactos se considerará como un gasto ajeno.

Tabla 18. Coste de Material y Hardware.

	Precio [€]
GPU Titan XP (12 GB)* de la estación de trabajo *Deepbox	**630**

9.2. Software

El programario informático utilizado (programas de conexión VPN o SSH, múdulos, librerías, etc.) es de *Open Source*, por lo tanto, gratuito.

9.3. Mano de obra e ingeniería

En cuanto a la mano de obra relativa a los procesos de extracción sanguínea e introducción de ésta en el microscópico electrónico, no se incluirá ya que se considerará un gasto ajeno a la investigación realizada.

Computarán como costes relativos a la ingeniería las horas trabajadas del autor del TFG junto con su director y co-tutor. Las horas del autor se calcularán en base al sueldo de un ingeniero biomédico junior (6 € la hora). Los sueldos de los directores se contarán como 18 euros la hora.

Tabla 19. Coste de mano de obra e ingeniería.

	Horas [h]	Precio/Hora [€/h]	Subtotal [€]
Autor	500	6	3000
Director	30	18	540
Co-director	30	18	540
IVA 21%			856,8
Total	560		4936,8

9.4. Presupuesto final

La suma total asciende al coste final de 5566,8€:

Tabla 20. Coste total.

	Precio [€]
Mano de Obra e Ingeniería	4936,8
Materiales y Hardware	630
Total	5566,8

Bibliografía

[1] Sjöstrand E., y Jönsson J. (2018). Cell image transformation using deep learning. Recuperado de http://lup.lub.lu.se/student-papers/record/8945302

[2] Perez, A. (2020). Unpaired image-to-image translation using cycle-consistent adversarial networks. *Repositorio de GitHub*. Recuperado de https://github.com/alejo-perez-upc-77/TFG/tree/master/Notebooks_Results_TFG. [Accedido: 2-en-2020].

[3] Deng E., Socher R., Fei-Fei L., Dong W., Li K., y L. Li. (2009). IEEE Conference on computer vision and pattern recognition (CVPR), 00, page 248-255.

[4] Alferez E. S. (2015). Methodology for automatic classification of atypical lymphoid cells from peripheral blood cell images (Tesis doctoral). Recuperado de http://hdl.handle.net/10803/308328.

[5] Merino A., Brugués R., García R., Kinder M., Torres F. y Escolar, G. (2011). Estudio comparativo de la morfología de sangre periférica analizada mediante el microscopio y el CellaVision DM96 en enfermedades hematológicas y no hematológicas. *Revista del Laboratorio Clínico*. vol. 4, pp. 3-14.

[6] Pansombut T., Wikaisuksakul S., Khongkraphan K., y Phon-on A. (2019). Convolutional neural networks for recognition of lymphoblast cell images, *Computational Intelligence and Neuroscience*. doi:10.1155/2019/7519603.

[7] Zhao J., Zhang M., Zhou Z., Chu J., y Cao F. (2016). Automatic detection and classification of leukocytes using convolutional neural networks. *Medical & Biological Engineering & Computing*. vol. 55, no. 8, pp. 1287–1301.

[8] Hu B., Tang Y., Chang E., Yubo F., Maode L., Yan X. (2017). Unsupervised learning for cell-level visual representation in histopathology images with generative adversarial networks. *IEEE Journal of Biomedical and Health Informatics*. PP. 10.1109/JBHI.2018.2852639.

[9] Boldú L., Merino A., Alférez S., et al. (2019). Automatic recognition of different types of acute leukaemia in peripheral blood by image analysis. *Journal of Clinical Pathology*. doi: 10.1136/jclinpath-2019-205949.

[10] Karras, T., Laine, S. & Aila, T. (2018). A style-based generator architecture for generative adversarial networks. CoRR. Recuperado de https://arxiv.org/abs/1812.04948.

[11] Brock A., Donahue J., Simonyan K. (2019). Large scale gan training for high fidelity natural image synthesis. *ICLR*. Recuperado de https://arxiv.org/abs/1809.11096.

[12] Jun-Yan Zhu et al. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. Recuperado de https://arxiv.org/pdf/1703.10593.

[13] Isola P., Zhu J.Y., Zhou T. y Efros A. (2016) Image-to-image translation with conditional adversarial networks. Recuperado de https://arxiv.org/pdf/1611.07004.

[14] OpenStax, Anatomy & Physiology. *OpenStax CNX*. (2016). Recuperado de http://cnx.org/contents/14fb4ad7-39a1-4eee-ab6e-3ef2482e3e22@8.24. [Accedido: 22-sep-2019].

[15] Regina B. Red blood cells (erythrocytes). *ThoughtCo*, (2019) Recuperado de http://thoughtco.com/red-blood-cells-373487. [Accedido: 22-sep-2019].

[16] Lumen. Platelets - *Boundless Anatomy and Physiology*. (2018) Recuperado de https://courses.lumenlearning.com/ boundless-ap/ chapter/platelets/ [Accedido: 22-sep-2019].

[17] White blood cell. *Encyclopædia Britan*nica, *inc*. (2019) Recuperado de https://www.britannica.com/science/white-blood-cell. [Accedido: 22-sep-2019].

[18] Ashton, N. (2013). Physiology of red and white blood cells. *Anaesthesia & Intensive Care Medicine*, 14(6), 261–266. doi:10.1016/j.mpaic.2013.03.001.

[19] Hall, G. et al. (2006). Tratado de fisiología médica. in Elsevier (Ed.), *Tratado de Fisiología Médica* (pp. 190–191). Barcelona: Elsevier Saunders.

[20] The editors of encyclopaedia britannica. lymphatic systemn - anatomy. (2019). Recuperado de https://www.britannica.com/science/lymphatic-system [Accedido: 31-oct-2019].

[21] Jagannathan - Bogdan M., Zon LI. Hematopoiesis. Development. (2013) 140:2463–7. doi:10.1242/dev.083147. [Accedido: 31-oct-2019].

[22] Sorenson tcb and rl. histology guide - Chapter 8 - Hematopoiesis. Recuperado de http://www.histologyguide.com/slidebox/08-hematopoiesis.html. [Accedido: 24-oct-2019].

[23] Schulman I., Pierce M., Lukens A., Currimbhoy Z. (1960). Studies on thrombopoiesis. I. A factor in normal human plasma required for platelet production; chronic thrombocytopenia due to its deficiency. Blood. 16: 943–57. PMID 14443744.

[24] NCBI. Leukopoiesis - NCBI. (1998). Recuperado de https://www.ncbi.nlm.nih.gov/mesh/68019891. [Accedido: 30-oct-2019].

[25] Hillman R. S., Ault K. A., Leporrier M. (2005) HMR. Hematology in clinical practice. 5e ed. (Robert S. Hillman, M.D., Kenneth A. Ault Md, Michel Leporrier, M.D., Henry M. Rinder Md, eds.). China: Copyright © 2011 by The McGraw-Hill Companies, Inc.; 2005. Recuperado de https://hemonc.mhmedical.com/content.aspx?bookid=1802&sectionid=124978364.

[26] Hematopoietic cytokines. Sigma Aldrich (2019) - https://www.sigmaaldrich.com/technical-documents/articles/biofiles/hematopoietic-cytokines.html#ref. [Accedido: 30-oct-2019].

[27] Merino, A. (2005) Manual de citología de sangre periférica. 1a ed. (Médica GA, ed.). Madrid: Grupo Acción Médica; 2005.9-10.

[28] Ferreri DA. (2007). Marginal - zone lymphoma. state of the art oncology in europe. Recuperado de http://www.startoncology.net/professional-area/marginal-zone-lymphoma/?lang=en. [Accedido: 22-oct-2019].

[29] Ribrag, DV. Linfoma folicular - Portal de información de enfermedades raras y medicamentos huérfanos. (2010). Recuperado de https://www.orpha.net/consor/cgibin/OC_Exp.php?Lng= ES&Expert= 545. [Accedido: 23-oct-2019].

[30] Barreras, J. I., Mintz, I.E. Beider, B. (2014) Fisiología del anillo de Waldeyer. *Revista FASO*, año 21, vol 2 [Accedido: 5-nov-2019].

[31] Mac millan cancer support. Mantle cell lymphoma. (2019) Recuperado de https://www.macmillan.org.uk/information-and-support/lymphoma/lymphoma-non-hodgkin/understanding-cancer/types-of-non-hodgkin-lymphoma/mantle-cell-lymphoma.html. [Accedido: 4-nov-2019].

[32] Cancer.net. Leukemia - Chronic Lymphocytic - CLL: Introduction. (2017) Cancer.Net Editorial Board. Recuperado de https://www.cancer.net/cancer-types/leukemia-chronic-lymphocytic-cll/introduction. [Accedido: 5-nov-2019].

[33] Zepeda, C. (1999). Mononucleosis Infecciosa y Síndromes Similares. Rev Med Hond 1999; 67:248-257.

[34] Adewoyin, A.S., Nwogoh, B. (2014) Peripheral blood film – a review. *Ann Ib Postgrad Med*.; 12(2):71–79.

[35] Russel, S. y Norvig, P. (2009). Inteligencia Artificial: Un Enfoque Moderno (3rd edición). p. 229.

[36] Stuart J. Russell, Peter Norvig (2010) Artificial Intelligence: A Modern Approach, Third Edition, Prentice Hall .

[37] Hinton, J. y Sejnowski, T. (1999). Unsupervised learning: Foundations of neural computation. *MIT Press*.

[38] Zhao Youcai, Huang Sheng, in Pollution Control and Resource Recovery (2017). Cap. 4.4.8, pp. 87-88. doi:10.1016/B978-0-12-811754-5.00005-1.

[39] Roman, V. (2019). "unsupervised machine learning: clustering analysis". *Medium*. [Accedido: 5-nov-2019].

[40] Foster D. (2019) Generative. Deep learning teaching machines to paint, write, compose and play. *O'Reilly*; Recuperado de http://oreilly.com/catalog/errata.csp?isbn=9781492041948.

[41] Rosenblatt F. The Perceptron (1958): A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:6. doi:10.1.1.335.3398.

[42] Sharma, S. (2017) What the hell is perceptron? Towards data science. Recuperado de https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53. [Accedido: 10-nov-2019].

[43] Goodfellow, I. Bengio, Y. Courville, A. (2016) Deep Learning. *The MIT Press* ©. p. 13. Recuperado de http://www.deeplearningbook.org. [Accedido: 12-nov-2019].

[44] Nielsen, M. A. (2015). Neural networks and deep learning, *Determination Press*. Recuperado de http://neuralnetworksanddeeplearning.com/chap4.html. [Accedido: 8-nov-2019].

[45] Vinhas, A. (2019). The magic behind the perceptron network - Neural Networks Series - Chapter 1, Part 1. Towards Data Science. Recuperado de https://towardsdatascience.com/the-magic-behind-the-perceptron-network-eaa461088367. [Accedido: 8-nov-2019].

[46] Howard, J. (Fastai). (2019). Lecture: Lesson 5: Back propagation; Accelerated SGD; Neural net from scratch. *University of San Francisco*.

[47] Sze, V. Chen, Y. H. Yang, T. J. Emer, J. (2017). Efficient processing of deep neural networks: A tutorial and survey. Recuperado de https://arxiv.org/abs/1703.09039.

[48] Krizhevsky, A. Sutshever, I. y Hinton, G. E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems (NIPS)*. doi: 10.1145/3065386.

[49] Convolutional neural networks for visual recognition. *Universidad de Stanford* Recuperado de http://cs231n.github.io/neural-networks-1/. [Accedido: 10-nov-2019].

[50] Clevert D.A. y Unterthiner T. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). doi:1511.07289 .

[51] Uniqtech. (2018). Understand the softmax function in minutes. data science bootcam - Medium. Recuperado de https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d. Published 2018. [Accedido: 19-nov-2019].

[52] Reed, R. MarksII, R. J. (1999) Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, *Massachussets Institute of Technology*. ISBN 978-0-262-18190-7. pp. 155-156.

[53] Bishop, C. M. (1995) Neural networks for pattern recognition. *Oxford University Press*. p. 39. ISBN 0198538642.

[54] Parmar, R. (2018) Common Loss functions in machine learning. Recuperado de https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23. [Accedido: 11-nov-2019]

[55] Gómez, R. (2019). Understanding categorical cross-entropy loss, binary cross-entropy loss, softmax loss, logistic loss, focal loss and all those confusing names. *Repositorio de Github*. Recuperado de https://gombru.github.io/2018/05/23/cross_entropy_loss/. [Accedido: 11-nov-2019].

[56] Chablani, M. (2017) Deep learning concepts — PART 1. Recuperado de https://towardsdatascience.com/deep-learning-concepts-part-1-ea0b14b234c8. [Accedido: 11-nov-2019].

[57] Rumelhart, D., Hinton, G. y Williams, R. (1986). Learning representations by back-propagating errors. Nature 323, 533–536. doi:10.1038/323533a0.

[58] Bachman, D. (2007), Advanced Calculus Demystified, New York: McGraw-Hill, ISBN 978-0-07-148121-2.

[59] Optimization: stochastic gradient descent. *UFLDL*. *University of Stanford*. Recuperado de http://deeplearning.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/. [Accedido: 13-nov-2019].

[60] Amini, A. Rus, D. Massachusetts Institute Of Technology, Adapted By M. Atarod/Science. Recuperado de https://www.sciencemag.org/news/2018/05/ai-researchers-allege-machine-learning-alchemy [Accedido: 13-nov-2019]

[61] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics, 4(5), 1–17. 296.

[62] Tieleman, T. Hinton. G. (2012) Online Class. Neural Networks for Machine Learning. *COURSERA*.

[63] Kingma, D. P. y Adam J. Ba. (2014). A method for stochastic optimization. Recuperado de https://arxiv.org/abs/1412.6980.

[64] Howard, J. (2019) *Fastai*. Lecture: Lesson 2: Data cleaning and production; SGD from scratch; Neural net from scratch. University of San Francisco.

[65] Google - Dive into deep learning. model selection, underfitting and overfitting. Recuperado de https://d2l.ai/chapter_multilayer-perceptrons/underfit-overfit.html. [Accedido: 13-nov-2019].

[66] Kasturi, S. N. (2019) Underfitting and overfitting in machine learning and how to deal with it towards data science. Recuperado de https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf. [Accedido: 13-nov-2019]

[67] Krogh, A. y Hertz. J. A. (1991) A simple weight decay can improve generalization. In Advances in neural information processing systems. (NIPS) pp. 950–957, 1992. Recuperado de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.1947&rep=rep1&type=pdf. [Accedido: 13-nov-2019].

[68] Srivastava, N. Hinton, G. Krizhevsky, A. Sutskever, I. y Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Recuperado de http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf.

[69] Wang, J. Perez, L. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning. Recuperado de https://arXiv:1712.04621v1.

[70] Luo, P. Wang, X. W. Shao, W. Peng, Z. (2019) Towards Understanding Regularization in Batch Normalization. Recuperado de https://arxiv.org/abs/1809.00846.

[71] Jiménez, R.V. (2018) Reconocimiento automático de células malignas en sangre periférica a partir de imágenes digitales y utilizando redes neuronales convolucionales. Recuperado de http://hdl.handle.net/2117/173676.

[72] Detection and tracking of pallets using a laser rangefinder and machine learning techniques - Scientific Figure on ResearchGate. (2017) Recuperado de https://www.researchgate.net/figure/An-example-of-convolution-operation-in-2D-2_fig3_324165524. [Accedido: 16-nov-2019].

[73] Mallick, S. Nayak, S. (2018) Number of parameters and tensor sizes in a convolutional neural network (CNN). *Learn OpenCV*. Recuperado de https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/. [Accedido: 16-nov-2019].

[74] Goodfellow, I. J. Pouget-Abadie, J. Mirza, M. Xu, B., Warde-Farley, D. Ozair, S. Courville, A. y Bengio, Y. (2014). Generative adversarial networks. NIPS 2014.

[75] Ratliff, L. J. Burden, S. A. y Sastry, S. S. (2013). Characterization and computation of local nash equilibria in continuous games. In communication, control, and computing (Allerton), 51st Annual Allerton Conference on, pp. 917–924. IEEE.

[76] Goodfellow, J. NIPS 2016 tutorial: Generative adversarial networks. Recuperado de https://arxiv.org/abs/1701.00160.

[77] Johnson, J. Alahi, A. Li. (2016) Perceptual losses for real-time style transfer and super-resolution. Recuperado de https://arxiv.org/abs/1603.08155.

[78] He, K. Zhang, X. Ren, S. y Sun, J. (2016) Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778. doi: 10.1109/CVPR.2016.90.

[79] Angiodysplasia detection and localization using deep convolutional neural networks - Scientific figure on researchGate. Recuperado de https://www.researchgate.net/figure/AlbuNet-34-uses-pre-trained-ResNet-34-as-an-encoder-It-is-different-from-TernausNet-in_fig5_324718560. [Accedido: 25-nov-2019].

[80] Zhu, J. Park, T. Isola, P. Efros, A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *Repositorio de GitHub*. Recuperado de https://github.com/junyanz/CycleGAN. [Accedido: 2-en-2020].

[81] Tsung-Yi, L. Goya, P. Ross, G. Priya, P. He, K. (2018) Focal Loss for Dense Object Detection. Recuperado de https://arxiv.org/pdf/1708.02002.pdf.

[82] Gugger, S. (2018), The 1cycle policy – *Fastai*. Recuperado de https://sgugger.github.io/the-1cycle-policy.html. [Accedido: 5-en-2020].

[83] Unión Europea, "Directiva 2012/19/UE del Parlamento Europeo y del Consejo, de 4 de julio de 2012, sobre residuos de aparatos eléctricos y electrónicos (RAEE)," D. Of. la Unión Eur., p. L 197/ 38-71, 2012.

[84] dhobbs (usuario de Stack Overflow). (2012) "List directory tree structure in python?" https://stackoverflow.com/questions/9727673/list-directory-tree-structure-in-python. [Accedido: 6-en-2020].

Anexo A. Fragmentos de código

A.1. Rutina de listado de archivos en árbol

```python
import os

def list_files(startpath):
    for root, dirs, files in os.walk(startpath):
        level=root.replace(startpath, '').count(os.sep)
        indent = ' ' * 4 * level
        print('{}{}/'.format(indent, os.path.basename(root)))
        subindent = ' ' * 4 * (level + 1)
        for f in files:
            print('{}{}/'.format(subindent, f))
```

Fragmento de código 8. Función "list_files", cuyo cometido es mostrar el árbol de directorios del Docker. Fuente: [84].

A.2. Ejemplos de *Presets* de entrenamiento de las GANs

```
----------------- Options ---------------
               batch_size: 1
                    beta1: 0.5
          checkpoints_dir: ./checkpoints
           continue_train: False
                crop_size: 256
                 dataroot:    [default: None]
          dataset_mode: unaligned
             direction: AtoB
           display_env: main
          display_freq: 400
            display_id: 1
         display_ncols: 4
           display_port: 8097
         display_server: http://localhost
        display_winsize: 256
                  epoch: latest
            epoch_count: 1
               gan_mode: lsgan
                gpu_ids: 0
              init_gain: 0.02
              init_type: normal
               input_nc: 3
                isTrain: True                              [default: None]
              lambda_A: 10.0
              lambda_B: 10.0
        lambda_identity: 0.5
              load_iter: 0                                 [default: 0]
              load_size: 286
                     lr: 0.0002
         lr_decay_iters: 50
              lr_policy: linear
       max_dataset_size: inf
                  model: cycle_gan
            n_layers_D: 3
                  name: CanRuti_Clinic_NoNormal          [default: experiment_name]
```

```
                      ndf: 64
                    netD: basic
                    netG: resnet_9blocks
                     ngf: 64
                   niter: 100
               niter_decay: 100
                no_dropout: True
                   no_flip: False
                   no_html: False
                      norm: instance
               num_threads: 4
                 output_nc: 3
                     phase: train
                pool_size: 50
               preprocess: resize_and_crop
               print_freq: 100
             save_by_iter: False
          save_epoch_freq: 5
         save_latest_freq: 5000
           serial_batches: False
                   suffix:
         update_html_freq: 1000
                  verbose: False
----------------- End -------------------
```

Fragmento de código 9. Preset *de parámetros para el entrenamiento de la CycleGAN en los dos datasets de entrenamiento de Can Ruti (Dataset Separate* y *el no separado por clases).*

```
----------------- Options ---------------
             aspect_ratio: 1.0
               batch_size: 1
          checkpoints_dir: ./checkpoints
                crop_size: 256
                 dataroot:  [default: None]
             dataset_mode: colorization
                direction: AtoB
          display_winsize: 256
                    epoch: 50                                   [default: latest]
                     eval: False
                  gpu_ids: 0
                init_gain: 0.02
                init_type: normal
                 input_nc: 1
                  isTrain: False                                [default: None]
                load_iter: 0                                    [default: 0]
                load_size: 256
         max_dataset_size: inf
                    model: colorization                         [default: test]
               n_layers_D: 3
                     name: Colorization_Clinic_CanRuti          [default:
experiment_name]
                      ndf: 64
                    netD: basic
                    netG: unet_256
                     ngf: 64
               no_dropout: False
                  no_flip: False
                     norm: batch
                    ntest: inf
                 num_test: 184                                  [default: 50]
              num_threads: 4
                output_nc: 2
```

```
                    phase: test
                preprocess: resize_and_crop
              results_dir: ./results/
            serial_batches: False
                    suffix:
                  verbose: False
----------------- End -------------------
```

Fragmento de código 10. Preset *de parámetros para el entrenamiento de la ColorizationGAN en el dataset de entrenamiento no separado por clases.*

A.3. Rutina de formación de *dataroots*

En el Fragmento de código 19 se muestra como se realiza la creación de los *dataroot* de las variantes fake del conjunto Can Ruti. Al generarse en la carpeta de resultados de la GAN transformadora, los nuevos archivos de imagen conservan el mismo nombre, pero se les añade una terminación diferente (en este caso 'fake_B_rgb'). En esta rutina se seleccionan estos nombres de archivo adicionándole el texto de las imágenes requeridas mediante una función de expresiones regulares y se copian a otras carpetas.

```
1.  path = 'directorio deonde se encuentran las imágenes Colorization de Can Ruti
    CanRuti'
2.
3.  variant = os.listdir('Carpeta de VARIANT_LYMPHOCYTE Original')
4.  atypical = os.listdir('Carpeta de ATYPICAL_LYMPHOCYTE Original')
5.  blast = os.listdir('Carpeta de BLAST Original')
6.
7.  def catcher(x): return re.findall('([a-zA-Z]+_+[\d]+)', x)[0]
8.
9.
10. variant_png = [catcher(i)+'_fake_B_rgb.png' for i in variant]
11. atypical_png = [catcher(i)+'_fake_B_rgb.png' for i in atypical]
12. blast_png = [catcher(i)+'_fake_B_rgb.png' for i in blast]
13.
14. for i in variant_png:
15.     copyfile(os.path.join(path,i), os.path.join("/directorio/VARIANT_LYMPHOCYTE
    ", i))
16.
17. for i in blast_png:
18.     copyfile(os.path.join(path,i), os.path.join("/directorio/BLAST", i))
19.
20. for i in atypical_png:
21.     copyfile(os.path.join(path,i), os.path.join("/directorio/ATYPICAL_LYMPHOCYT
    E", i))
22.
23.
24. folder_1 = os.listdir("Carpeta de VARIANT_LYMPHOCYTE Original")
25. folder_2 = os.listdir("Carpeta de ATYPICAL_LYMPHOCYTE Original")
26. folder_3 = os.listdir("Carpeta de BLAST Original")
27. dic_Classif = {
28.       'ATYPICAL_LYMPHOCYTE':len(folder_1),
29.       'VARIANT_LYMPHOCYTE':len(folder_2),
30.       'BLAST':len(folder_3)
31.      }
```

```
32.
33. pd.DataFrame(dic_Classif, index=['N'])
```

Fragmento de código 19. Rutina de selección de imágenes fake para crear los dataroots del conjunto Can Ruti Colorization.

A.4. Rutina de separación de archivos entre *training* - *validation set* por *dataframe* (Conjunto Can Ruti)

Para utilizar el mismo *training set y validation set e*n los conjuntos Can Ruti originales y Can Ruti fake se realiza lo siguiente. Primero se exportan dos *csv* de las rutas de los archivos incluidos en cada *set* del ensayo con el conjunto Can Ruti original (uno para *training* y otro para *validation*). Después se leen estos *csv* (en *dataframe)* y se le añaden los cambios de nombres de los archivos generados como en el punto anterior. Por último, se concatenan los dataframes (con columnas anexas con información de etiqueta y de si pertenecen al *training* o a *validation set*). Finalmente se incluye este *dataframe* en el código de DataBlock API explicitando la separación mediante *dataframe.*

```
1.  df_train = pd.read_csv('/shared/can_ruti/train.csv')
2.  df_train['x'] = df_train['x'].apply(lambda x: x.replace('.jpg', '_fake_B_rgb.png')
    )
3.  df_train['z'] = False
4.
5.  df_valid = pd.read_csv('/shared/can_ruti/validation.csv')
6.  df_valid['x'] = df_valid['x'].apply(lambda x: x.replace('.jpg', '_fake_B_rgb.png')
    )
7.  df_valid['z'] = True
8.
9.  df_split = pd.concat([df_train, df_valid])
10. df_split = df_split.rename(columns={'x': 'name','y': 'label','z':'is_valid'})
11.
12. data = (ImageList.from_df(df_split, folder_classes)
13.         .split_from_df()
14.         .label_from_folder()
15.         .transform(tfms, size=224)
16.         .databunch()
17.         .normalize(imagenet_stats))
```

Fragmento de código 20. Creación de separaciones train-validation iguales para todos los datasets a clasificar del conjunto Can Ruti.

Anexo B. Versión extendida de resultados

B.1. Ejemplos de matrices de confusión del conjunto Can Ruti antes y después del fine tuning en los respectivos datasets (Bloque 2)

Tabla 21. Matriz de confusión normalizada ResNet_34_No_SIND (Can Ruti Original) antes del Fine Tuning en el conjunto Can Ruti Original.

Confusion Matrix			
Atypical Lymphocyte	1.00	0.00	0.00
Blast	0.84	0.16	0.00
Variant Lymphocyte	0.97	0.00	0.03
	Atypical Lymphocyte	Blast	Variant Lymphocyte

Actual

Predicted

Tabla 22. Matriz de confusión normalizada. ResNet_34_No_SIND (Can Ruti Original) después del Fine Tuning en el conjunto Can Ruti Original.

Confusion Matrix			
Atypical Lymphocyte	0.96	0.00	0.04
Blast	0.00	1.00	0.00
Variant Lymphocyte	0.00	0.00	1.00
	Atypical Lymphocyte	Blast	Variant Lymphocyte

Actual

Predicted

Tabla 23. Matriz de confusión normalizada. ResNet_34_No_SIND (Can Ruti CycleGAN 15 Epochs No Separate) antes del Fine Tuning en el conjunto Can Ruti CycleGAN 15 Epochs No Separate.

Confusion Matrix			
Atypical Lymphocyte	0.96	0.04	0.00
Blast	0.10	0.86	0.00
Variant Lymphocyte	0.47	0.10	0.43
	Atypical Lymphocyte	Blast	Variant Lymphocyte

Actual

Predicted

Tabla 24. Matriz de confusión normalizada. ResNet_34_No_SIND (Can Ruti CycleGAN 15 Epochs No Separate) después del Fine Tuning en el conjunto Can Ruti CycleGAN 15 Epochs No Separate.

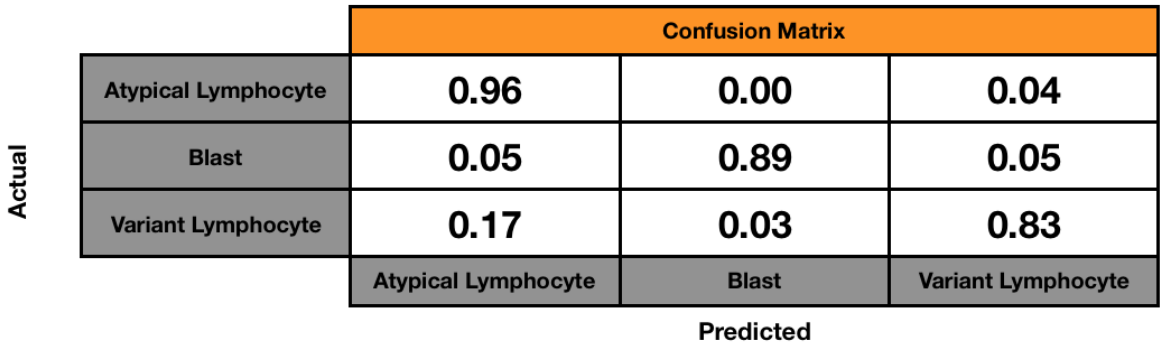


Tabla 25. Matriz de confusión normalizada. ResNet_34_No_SIND (Can Ruti CycleGAN 20 Epochs No Separate) antes del Fine Tuning en el conjunto Can Ruti CycleGAN 20 Epochs No Separate.

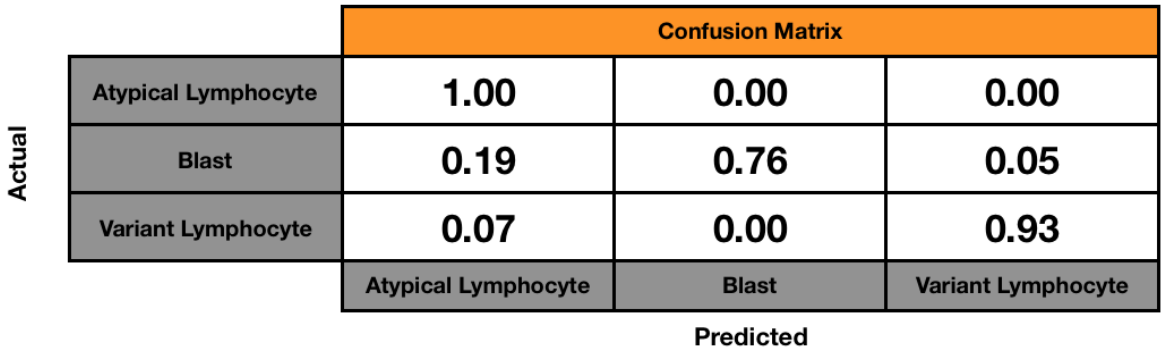


Tabla 26. Matriz de confusión normalizada. ResNet_34_No_SIND (Can Ruti CycleGAN 20 Epochs No Separate) después del Fine Tuning en el conjunto Can Ruti CycleGAN 20 Epochs No Separate.

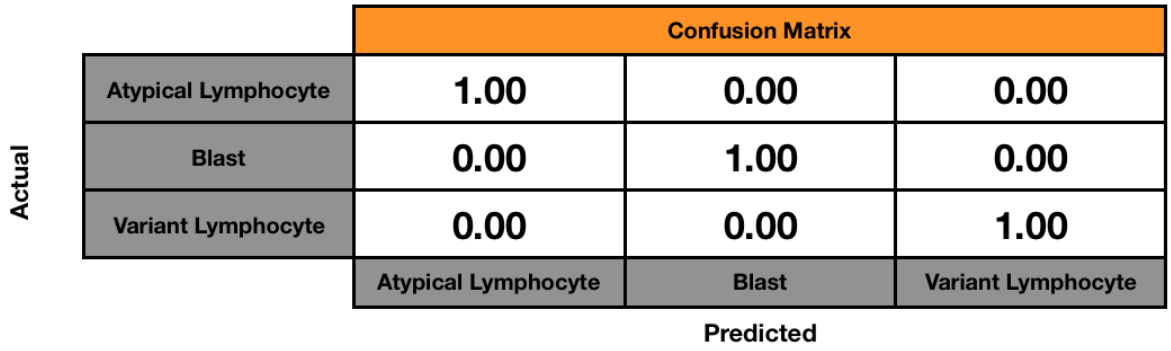


Tabla 27. Matriz de confusión normalizada. ResNet_34_Focal (Can Ruti CycleGAN Epochs Óptimos Separate Dataset) antes del Fine Tuning en el conjunto Can Ruti CycleGAN Epochs Óptimos Separate Dataset.

		Confusion Matrix		
Actual	Atypical Lymphocyte	1.00	0.00	0.00
	Blast	0.41	0.59	0.00
	Variant Lymphocyte	0.87	0.13	0.00
		Atypical Lymphocyte	Blast	Variant Lymphocyte
		Predicted		

Tabla 28. Matriz de confusión normalizada. ResNet_34_Focal (Can Ruti CycleGAN Epochs Óptimos Separate Dataset) después del Fine Tuning en el conjunto Can Ruti CycleGAN Epochs Óptimos Separate Dataset.

		Confusion Matrix		
Actual	Atypical Lymphocyte	0.92	0.00	0.00
	Blast	0.00	1.00	0.00
	Variant Lymphocyte	0.00	0.03	0.97
		Atypical Lymphocyte	Blast	Variant Lymphocyte
		Predicted		

B.2. resultados extendidos de transformación (Conjunto Can Ruti, Bloque 1

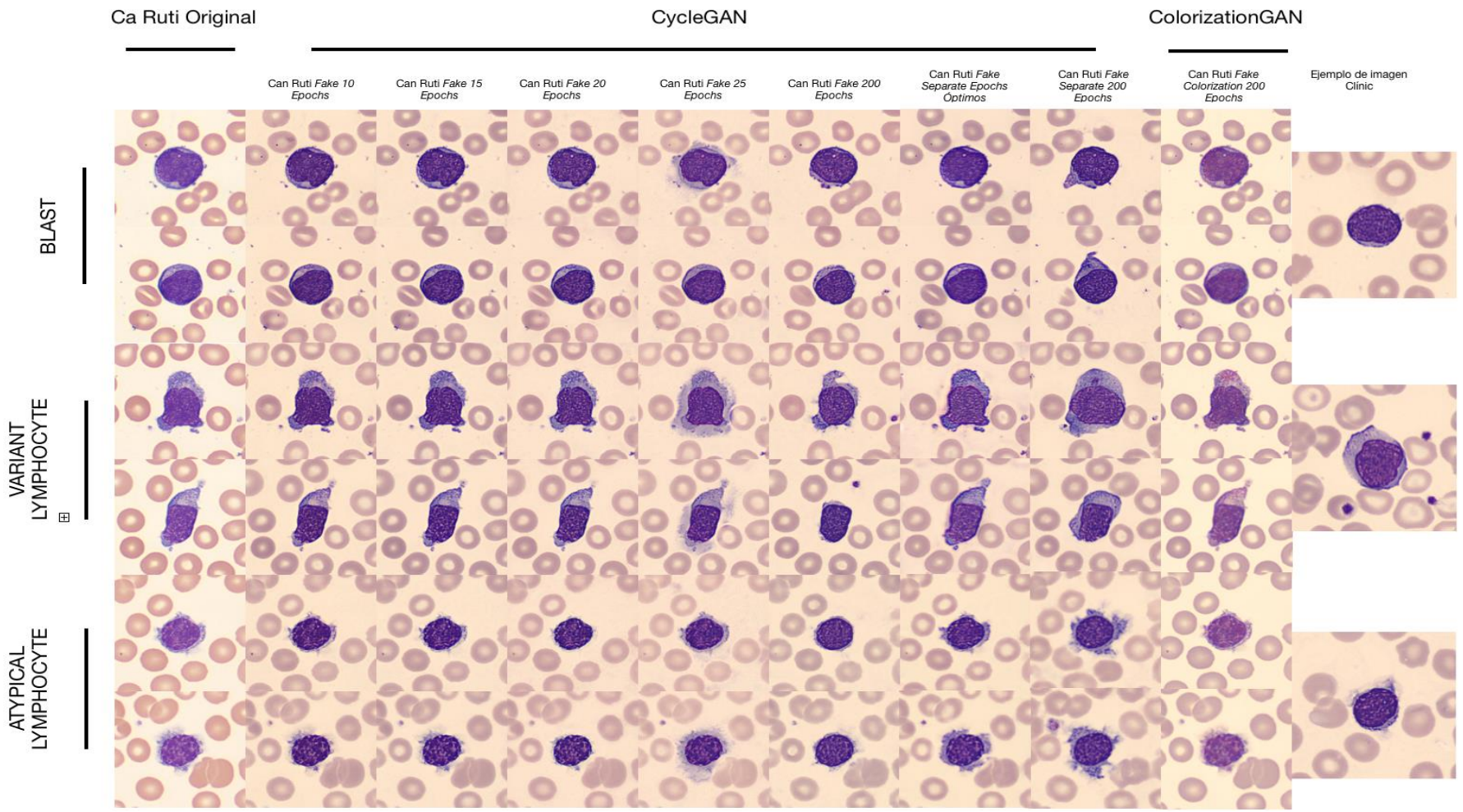


Figura B.2. Ejemplos extendidos de los resultados de transformación del conjunto Can Ruti (Bloque 1). Fuente: Elaboración Propia.